מבוא לתכנות בסביבת אינטרנט

לפי תוכנית הלימודים של משרד החינוך

מהדורה ניסויית

ספר זה בנוי על פי תוכנית הלימודים של משרד החינוך, האגף למדע וטכנולוגיות, מדעי המחשב ומגמת טכנולוגיות מידע. תוכנית זו, בהיקף של 90 שעות לימוד עיוניות ומעשיות, מיועדת לתלמידים הלומדים מדעי המחשב ברמה הרגילה שסיימו יסודות 1.

פנה לאתר האינטרנט של הספר כדי להוריד את קוד המקור של כל התוכניות, חומר נוסף ושאלות תרגול לספר זה:

www.hod-ami.co.il/59316.html

עורך ראשי: **זהר עמיהוד** 

מרכז תוכנית beta: **צור ריכטר-לוין** 

עריכה ועיצוב: **טליה טופז**

עיצוב עטיפה: **שרון רז**

תודות:

תודה מיוחדת ל- **ד״ר אבי כהן**, המפמ״ר למדעי המחשב, שכתב את התוכנית והציג את הדרישות המקצועיות הפדגוגיות של הוראת הנושא.

אנו מודים למורים ולמורות שהשתתפו בתוכנית ה-beta של ספר זה.

הערותיכם מאפשרות לנו להגיש לקהל הלומדים את המוצר הטוב ביותר האפשרי, ומאפשרות לתלמידים לעמוד בדרישות תוכנית הלימודים באופן הטוב ביותר.

תגובותיכם והערותיכם היו חשובות לנו מאוד ונעזרנו בהן רבות במהלך פיתוח חומר הלימוד בספר זה.

תודה מיוחדת למורים ולמורות (הרשימה מסודרת לפי א״ב):

איזבלה טבלין
שבח מופת, תל אביב

אילן פרץ
מכללת אורט ע״ש יוסף חרמץ, ירושלים

אילנה גורודצקי
מקיף ע״ש יצחק רגר, באר שבע

אלה לב
גימנסיה ריאלית ע״ש קררי, ראשון לציון

ורד בראון
רנה קאסין, ירושלים

טובי סטפ
עמל ב׳, פתח תקוה

ישראל יוסף
תיכון חדש, תל אביב

נורית פינטל
תיכון בית רבקה

עופר סובול
העמק המערבי, קיבוץ יפעת

עמיחי פיגנבוים
אוניברסיטת בר-אילן, נוער שוחר מדע

ראובן יגל
קריה חינוכית, מצפה רמון

שמעון אייבס
באר שבע

מבוא לתכנות
בסביבת אינטרנט

לפי תוכנית הלימודים
של משרד החינוך

מהדורה ניסויית

זהר עמיהוד

ירון לייפנברג

אייל לייפנברג



הוצאת הוד-עמי לספרי מחשבים בע"מ

ת.ד. 6108 הרצליה 46160

טלפון: 09-9564716 פקס: 09-9571582

www.hod-ami.co.il

info@hod-ami.co.il

לשם שטף הקריאה כתוב ספר זה בלשון זכר בלבד.
ספר זה מיועד לגברים ונשים כאחד
ואין בכוונתנו להפלות או לפגוע בציבור המשתמשים/ות.

מהדורה ראשונה 2001

HOD-AMI Ltd.
P.O.B. 6108, Herzliya
ISRAEL, 2001

ISBN 965-361-285-9 מסת"ב

תוכן עניינים מקוצר

חלק 4 - ASP

תוכן העניינים

חלק 1 - מבוא

חלק 2 - HTML

חלק 3 - JavaScript

חלק 4 - ASP

הקדמה

מבוא לתכנות בסביבת אינטרנט

ספר זה בנוי על פי תוכנית הלימודים של משרד החינוך, האגף למדע וטכנולוגיות, מדעי המחשב ומגמת טכנולוגיות מידע. תוכנית זו, בהיקף של 90 שעות לימוד עיוניות ומעשיות, מיועדת לתלמידים הלומדים מדעי המחשב ברמה הרגילה שסיימו יסודות 1.

ספר זה מכסה את כל תכני הלימוד כפי שמופיעים בתוכנית הלימודים. תלמיד שילמד לפי ספר זה יוכל לעמוד בדרישות הקורס כפי שהוגדרו :

- הכרת עקרונות העברת המידע באינטרנט.
- הכרת המושגים הבסיסיים של תגיות HTML.
- הכרת המושגים הבסיסיים של תכנות תסריטים עם JavaScript.
- הכרת המונחים הבסיסיים של תכנות מבוסס אובייקטים.
- הכרת המושגים הבסיסיים של ASP.
- הכרת המושגים הבסיסיים של תכנות בצד הלקוח ובצד השרת.
- יכולת ליצור אתרים אינטראקטיביים מבוססי תסריטים בסיסיים.

התכנים של ספר זה, כאמור, מכסים את התכנים ורשימת הפקודות, האובייקטים, השיטות והמאפיינים הנדרשים על פי תוכנית הלימוד. חומר עזר נוסף בנושאים אלה, ניתן למצוא בספרים הבאים :

- **HTML 4 למפתחי אתרים באינטרנט**, מהדורה שלישית, זהר עמיהוד, הוצאת **הוד-עמי**.
- **JavaScript למפתחי אתרים באינטרנט**, ירון לייפנברג, הוצאת **הוד-עמי**.
- **JavaScript המדריך השלם**, הוצאת **הוד-עמי**.
- **ASP 3 המדריך השלם**, Stephen Walther , הוצאת **הוד-עמי**.
- **ASP 3 סדנת לימוד**, כולל Visual InterDev, רונן אלמוג, הוצאת **הוד-עמי**.
- **ASP 3 למפתחי אתרים באינטרנט**, ירון ואייל לייפנברג, הוצאת **הוד-עמי**.

כמו כן, מומלץ לעיין בספר **עיצוב ממשק באינטרנט**, Jackob Nielsen, הוצאת **הוד-עמי**.

ספרות בנושא מערכת הפעלה Windows בהוצאת **הוד-עמי**:

Windows 95 תכל'ס, כ- 230 עמודים + תקליטון

Windows 95 הסדרה הידידותית, 528 עמודים

Windows 98 תכל'ס, כ- 240 עמודים + תקליטון

Windows 98 קוראים יודעים, כ- 592 עמודים + תקליטור

Windows 2000 קוראים יודעים, כ- 512 עמודים + תקליטור

Windows Me הסדרה הידידותית למתחילים, כ- 144 עמודים

תוכן עניינים ופרק לדוגמה של כל ספר וספר ניתן למצוא באתר הוצאת הוד-עמי באינטרנט בכתובת **www.hod-ami.co.il**.

כיצד בנוי הספר?

הספר בנוי ממספר חלקים:

מבוא - הסבר קצר ועינייני על התפתחות רשת האינטרנט ועל TCP/IP בצד מושגים שכיחים. חלק קצר זה גם יעסוק בנושא כתובת IP.

HTML - לימוד התגיות העיקריות בדף HTML להצגת תכנים (טקסט ותמונות) הכוללים: צבע, קישורים, טבלאות, טפסים ומסגרות.

JavaScript - לימוד שפת תסריט (script) דרך מעבר קצר על התחביר, פונקציות עיקריות, טיפול בטפסים וחלונות.

ASP - עבודה מול שרת. הסבר עקרוני על דרך העברת המידע ברשת, דרך הכרת האובייקטים העיקריים וגם עבודה עם קבצים.

תוכנות נדרשות

שלב לימוד	תוכנה נדרשת (דרישות מינימום)
מבוא	-
HTML	עורך טקסט פשוט כמו פנקס רשימות (Notepad). התוכנה נמצאת כחלק מ-Windows.
JavaScript	דפדפן Microsoft Internet Explorer. רצוי בגירסה מעודכנת.
ASP	תוכנת Microsoft PWS.

עורך טקסט מינימלי הוא עורך טקסט פשוט כמו **פנקס רשימות** (Notepad). ניתן גם להשתמש ב-Word או ב**כתבן** (WordPad) אבל יש לשמור את הקובץ כקובץ **טקסט** (txt) ולא כקובץ doc. באינטרנט ניתן למצוא עורכי טקסט משוכללים יותר המותאמים לעבודה עם HTML.

דפדפן Microsoft Internet Explorer בגירסה מעודכנת. ספר זה נבדק על ידי גירסה 5 של הדפדפן. פקודות ומאפיינים מסוימים כמו גם אובייקטים ושיטות עשויים שלא לפעול בגרסאות ישנות יותר (ובמיוחד גירסה 3 ומטה).

ASP 3 נתמכת במלואה רק בשרת האינטרנט של Microsoft בגירסה 5 (IIS 5). אבל, הפעלתו של שרת זה אפשרית רק למי שברשותו Windows NT 4 או Windows 2000 Server. מכיון שכך, חברת Microsoft שחררה מוצר הנקרא PWS (Personal Web Server) הניתן להתקנה ב-Windows 95/98.

היכן ניתן למצוא תוכנות אלו?

עורך טקסט

עורך טקסט (text editor) כמו תוכנת **Notepad** (הנקראת פנקס רשימות) שנמצאת בכל ערכת הפעלה Windows :

פתח את תפריט **התחל** (Start), **תוכניות** (Programs), **עזרים** (Accessories), **פנקס רשימות** (Notepad).

למי שמעוניין, יש מיגוון גדול של עורכי טקסט משוכללים יותר, freeware או shareware, אותם ניתן למצוא באתר www.tucows.co.il ובאלפי אתרים אחרים ברחבי האינטרנט.

תוכנת Internet Explorer בגירסה מעודכנת

Windows 95/98/ME : ניתן לבצע שדרוג הגירסה באמצעות האינטרנט.

תוכנת Internet Explorer בגירסה מעודכנת נמצאת בתקליטור המצורף כמעט לכל ספר בהוצאת הוד-עמי (הגירסה מעודכנת ליום הדפסת התקליטור).

PWS

את תוכנת PWS ניתן למצוא בתקליטורים המצורפים לספרים הבאים:

- **HTML 4 למפתחי אתרים באינטרנט**, מהדורה שלישית, זהר עמיהוד, הוצאת **הוד-עמי**.
- **JavaScript למפתחי אתרים באינטרנט**, ירון לייפנברג, הוצאת **הוד-עמי**.
- **JavaScript המדריך השלם**, הוצאת **הוד-עמי**.
- **ASP 3 המדריך השלם**, Stephen Walther , הוצאת **הוד-עמי**.
- **ASP 3 סדנת לימוד**, כולל Visual InterDev, רונן אלמוג, הוצאת **הוד-עמי**.
- **ASP 3 למפתחי אתרים באינטרנט**, ירון ואייל לייפנברג, הוצאת **הוד-עמי**.

Windows 95: התוכנה לא נמצאת בתקליטור ההתקנה. נדרש לבצע עדכון ל-Winsock 2.0.

Windows 98: התוכנה נמצאת בתקליטור ההתקנה בתיקיה \add-ons\pws.

Windows ME: התוכנה לא נמצאת בתקליטור ההתקנה וגם לא ניתן להתקין אותה במערכת הפעלה זו.

סימנים מוסכמים בספר

ספר זה מכיל מספר מוסכמות ואלמנטים שמטרתם לסייע לך במציאת מידע במהירות.

הערה

תחת הכותרת הערה תוכל למצוא מידע נוסף.

טיפ

תחת הכותרת טיפ תוכל למצוא מידע נוסף הנוגע לנושא הנדון.

אזהרה

האזהרות משמשות כתמרור המורה לך שקיימת אפשרות שתיתקל בבעיה, ומנחה אותך כיצד להימנע ממנה.

שאלה ותשובה!

תחת הכותרת שאלה ותשובה תמצא בעיקר מידע אודות תקלות אפשריות ופתרונן.

בעברית פשוטה!

נתקלת במונח או מושג חדש? תוכל למצוא הסבר תחת כותרת זו.

תחביר, תחביר ועוד פעם תחביר

שפת HTML וכך גם JavaScript ו-VBScript כולם הינן "שפות חופשיות", במידה זו או אחרת. אם יש לך רקע של שפות תכנות כמו Pascal, C, C++ או Java בהן יש חוקים נוקשים של תחביר - כאן הדברים שונים. שפות התסריט JavaScript ו-VBScript הן שפות הנקראות Loosely Type Languages. מצד אחד זה "כייף" שניתן להשתמש במשתנה בלי להגדיר אותו, אבל מצד שני זה עשוי לבלבל ולהוביל לתוצאות "לא צפויות/שגויות" כי מישהו אחר (במקרה זה הדפדפן, ה-script engine) עשה את העבודה עבורך ו... הוא יכול גם לטעות!

מעבר לכך, שפות כמו Pascal או C הן שפות "מבוססות" כבר שלושים פלוס, לעומת שפות התסריט שקיימות לא יותר מכמה שנים. בשנותיהן הקצרות, עברו שפות התסריט הרבה מאוד גרסאות ועדיין "לא התבססו".

מכיון שרשת האינטרנט היא כל כך דינמית, שפות התסריט לא רק שצריכות להיות תואמות לאחור, הן לא מצליחות להתאים לתקנים המתפרסמים חדשות לבקרים: XML, ASP.NET, WAP, XHTML ועוד.

בספר זה נתעקש על כתיבת תחביר נכון. למרות שניתן לכתוב אחרת ונראה שניתן לחסוך זמן ומאמץ, שווה לבדוק בכללי כתיבת קוד נכונים גם בשפות "חופשיות ומשוחררות" כמו JavaScript ו-VBScript. "החיסכון", לכאורה, יתבטל ברגע הראשון שתופיע איזו שהיא בעיה, שתדרוש התאמה לקוד.

World Wide Web היא רשת דינמית

יש לזכור תמיד, כי הרשת העולמית חיה ודינמית. אתרים חדשים נוספים לרשת בקצב של מאות ליום, בעוד שרבים אחרים גוועים ונעלמים. גם אתרים גדולים ומכובדים משנים את כתובתם מפעם לפעם.

הספר שבידך מכיל כתובות רבות ברשת. **כתובות אלו נבדקו בקפידה, אולם לא ניתן לעקוב אחר השינויים המתמידים של האינטרנט**. כך, שאם אתה מוצא עצמך מול כתובת כגון: Site was not found, או Document Not Found - אל ייאוש. בדוק את הכתובת שהקלדת, ונסה אותה בשינויים קלים. לעיתים קרובות תראה שינויים כאלה:

www.a_company_name.com/ ➔ www.here.com/a_company_name/

www.XYZ.com/ ➔ www.XYZ_company.com/

אך כאשר אתה תערוך שינויים באתר שתבנה בעתיד הקרוב, זכור להשאיר עקבות ברורים שינחו את קוראיך לכתובתך החדשה!

שים לב!

ספר זה מציג את תהליכי ההפעלה והכוונון של תוכנות בסביבת מערכת ההפעלה Windows 95/98. Windows 9x מתאימה את המסכים לתצורת החומרה והתוכנה שמותקנים במחשב שלך. לכן, ייתכן שהמסכים שתראה במסך שבמחשב שלך יהיו שונים מעט מאלה שמתוארים בספר.

הערה חשובה!

הכתובות והמסכים בספר זה נאספו במהלך חודש אפריל/מאי 2001. רשת האינטרנט מאוד דינמית, כפי שתיווכח במהלך העבודה, ולכן בהחלט ייתכן:

שמסכים שתראה על המסך שלך יהיו שונים מאלה המובאים בספר.

שכתובות לא תהיינה נכונות מאחר והדף הורד או שהמחשב נסגר.

הוצאת **הוד-עמי** עשתה, עושה ותעשה כל מאמץ להביא לפניך את החומר העדכני ביותר. לכן, השתדלנו שהכתובות המוזכרות בספר יהיו במחשבים שימשיכו לתת שירותי אינטרנט עוד שנים רבות.

הערה נוספת חשובה!

הסיומות HTM ו-HTML זהות.

עברית באינטרנט

מה לעשות, עברית אינה השפה הרשמית של האינטרנט. עברית כותבים מימין לשמאל כאשר עולם האינטרנט הפוך! לעברית יש אותיות (גופנים) שאין בשפה אחרת ו... חברות Microsoft ו-Netscape פתחו את הדפדפנים שלהם שיציגו מסמכים ב... אנגלית.

אם כך, מה עושים כדי לראות ולעבוד בעברית באינטרנט? ובכן, עד כה פותחו שלוש גישות עיקריות להצגת העברית: Logical (על ידי חברת Microsoft), Semi-Logical ו-Visual (סנונית) וזוהי רק ההתחלה.

דפדפן Internet Explorer תומך בעברית במקור, ולכן אין צורך בהתקנות גופנים למיניהם. בעת התקנת דפדפן Microsoft Internet Explorer יש לבחור בתמיכה בהצגת השפה העברית. לפרטים נוספים ראה בספר **אינטרנט עם Intenet Explorer 5.x קוראים יודעים**.

חייבים לדעת HTML כדי לכתוב עמודי HTML בעברית

המשמעות היא שצריך לקרוא, ללמוד "וללכלך את הידיים" עם HTML (באנגלית) לפי ספר זה כדי לבנות אתר (באנגלית), ורק לאחר שמבינים ומיישמים (באנגלית), אפשר לגשת ולבנות אתר **בעברית**. כתיבת דפי HTML בשפה העברית מוסברת בפרק 9.

במי לבחור Internet Explorer או Netscape?

עם כל הכבוד ליוצרי Netscape שהיו חלוצים בתחומם - העברית היא לא הצד החזק של תוכנה זו. חברת Microsoft עם Internet Explorer עשתה עבודה טובה בכל הנוגע לעברית באינטרנט. לכן, מרבית הדוגמאות שתראה בספר זה הורצו על גרסאות 4 ו-5 של תוכנת Microsoft Internet Explorer, מכיון שלכתוב דפי HTML בעברית זה קשה, אבל אם הם מוצגים בעזרת דפדפן Internet Explorer - קל לכתוב וקל לגלוש. שים לב, דף שנכתב בעברית לוגית לצפייה בדפדפן Internet Explorer לא ייראה טוב עם תוכנת Netscape. דע כי יש דרך לכתוב דף שיהיה טוב לצפייה עם Internet Explorer וגם עם Netscape אבל אתה תצטרך לעבוד, ולעבוד קשה. ובכלל לא בטוח שזה כדאי, מפני שנכון לעכשיו למעלה מ- 88% מהגולשים בעולם משתמשים ב-IE.

לא רואה עברית

לפעמים, בשל אופן כתיבת האתר (ולא ניכנס לפרטים הטכניים בנושא), Internet Explorer אינו מצליח לזהות את העברית באופן אוטומטי.

אם נכנסת לאתר והעברית בו אינה מוצגת כהלכה, אין בעיה!

1. לחץ לחיצה ימנית על נקודה ריקה כלשהי בדף בו אתה צופה.
2. בתפריט הקיצור הצבע על **קידוד** (Encoding).
3. בחר באפיון העברית הנדרש עבור האתר.

אינך רואה אפיון שפה כלשהו, או שהאפיון המבוקש אינו מופיע בתפריט? גם כאן יש פתרון: בתפריט המשנה של **קידוד** (Encoding) הצבע על **יותר** ותוכל לבחור אפיון אחר.

האפשרויות העומדות בפניך (בעברית) הן: **(DOS) עברית**, **(ISO-Logical) עברית**, **(ISO-Visual) עברית** או **(Windows) עברית**. פעל בשיטת הניסוי והטעייה כדי לבחון את אפיון העברית המתאים לאתר בו הינך צופה.



בחלקו התחתון של התפריט מופיעות שתי אפשרויות: **מסמך משמאל-לימין** או **מסמך מימין-לשמאל**. אם גלשת לאתר בעברית שאינו מוצג נכון במסך (צמוד כולו לשמאל, במקום לימין) לחץ לחיצה ימנית במקום ריק כלשהו בדף בו אתה צופה, בתפריט הקיצור הצבע על **קידוד** וסמן בו את האפשרות **מסמך מימין-לשמאל**.

קבצי קוד המקור

לספר זה לא מצורף תקליטור.

את קוד המקור של כל הדוגמאות בספר ניתן להוריד מאתר הוד-עמי בכתובת:

www.hod-ami.co.il/59316.html

שאלות, הערות, רעיונות

לא נחסך כל מאמץ להבטחת הדיוק של ספר זה וקבצי המקור הנלווים אליו. אם יש לך הערות, שאלות או רעיונות הנוגעים לספר זה של ההוצאה, אנא שלח אותם להוצאת הוד-עמי באחת השיטות הבאות:

דואר אלקטרוני: support@hod-ami.co.il , בשורת Subject ציין את המספר 59316

דואר רגיל:

הוצאת הוד-עמי לספרי מחשבים
ת.ד. 6108
הרצליה 46160

חלק 1 - מבוא

מבוא

ראשית האינטרנט

רשת האינטרנט העולמית - the global Internet - המקיפה של היום, החלה כמערכת צנועה למדי. בשנת 1969 פיתחה **סוכנות ARPA** (Advanced Research Projects Agency) של משרד ההגנה האמריקאי רשת ניסיונית שנקראה ARPAnet - לקישור ארבעה מרכזי מחשוב-על למטרות מחקר צבאי. דרישות התכנון של רשת זו היו מהירות, אמינות וסיבולת למקרה שפצצה גרעינית תהרוס את כל אחד ממרכזי המחשוב ברשת. מאותם ארבעה מחשבים מקוריים, הרשת התפתחה לרשת המשתרעת על מיליוני מחשבים שאנו מכירים כיום, רשת **האינטרנט**.

למרות שאיום הפצצה הגרעינית פחת כיום, התברר שתכנון זה הינו נכס עצום לרשת תקשורת מרחבית. כל מחשב אחראי להעביר את המידע שלו למחשב אחר, ולרשת עצמה אין אחריות כלשהי. אם מחשב או חיבור למחשב כושלים מסיבה כלשהי, התקשורת למחשב זה תיפסק, אולם שאר המחשבים ימשיכו לפעול ולתקשר ביניהם. הרשת לא תיפול בגלל מחשב אחד.

התרומה הגדולה ביותר של ARPAnet היא כנראה פיתוח מערכת הפרוטוקולים TCP/IP. כאשר מערכת זו הפכה למערכת הפרוטוקולים התקנית ברשתות, היא איפשרה למחשבים מכל הסוגים להיות מחוברים זה לזה ולשתף ביניהם מידע.

במשך הזמן, ARPAnet דעכה והוחלפה ברשת NSFnet, שגם היא היתה במימון ממשלתי והופעלה בחסות **NSF** (National Science Foundation). גם NSFnet "סיימה את תפקידה" והוחלפה על ידי **סריג** (mesh) של רשתות מסחריות שאנו מכירים היום. ממשלת ארה"ב עדיין מממנת חלקים של האינטרנט המוקדשים לתחומים ממשלתיים, צבאיים וחינוכיים, למרות שכיום האינטרנט היא בעיקרה מפעל מסחרי.

למעשה, האינטרנט עצמה היא "**רשת של רשתות**". אין כל "חברת אינטרנט" מרכזית שאליה ניתן להתחבר. לפנינו אוסף של **ספקי שירותי אינטרנט - ISP** (Internet Service Providers) המפעילים את הרשתות שלהם, עם הלקוחות שלהם, ומסכימים ביניהם להתחבר זה לזה, כדי להעביר מנות נתונים מרשת אחת לאחרת. ספקי שירותי אינטרנט גדולים רבים מוכרים לספקים קטנים יותר את אפשרות החיבור לרשת שלהם, וחלקם מוכרים חיבורים לספקי שירות אחרים.

בסופו של דבר, ספקי שירותים אלה בכל הרמות מוכרים חיבורים לבודדים ולחברות, המשלבים את הרשתות (או המחשבים הבודדים) שלהם לרשת גדולה יותר.

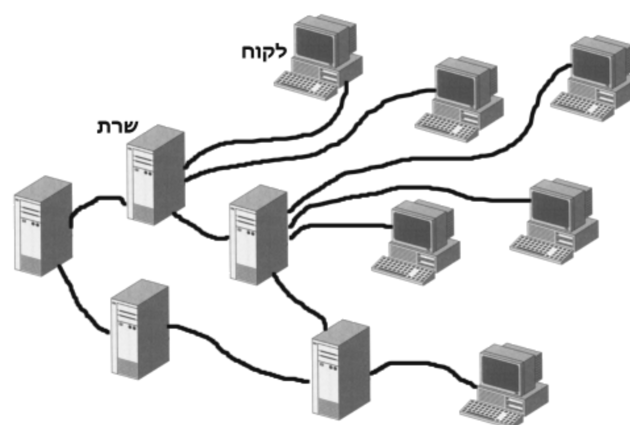
למרות שאין שליטה מרכזית של האינטרנט, קיימים תקני תקשורת ויש תיאום בין ספקי השירותים. הדבר נעשה בפיקוח ארגון **מלכ"ר** (מוסד ללא כוונת רווח) הנקרא Internet Society. ארגון נוסף כזה הוא **IETF** (Internet Engineering Task Force), שמתאם את הפעולות של מספר ועדות המגדירות תקני תקשורת אינטרנט וחוקרות שיטות להרחבה ושיפור תקשורת אינטרנט. תקני התקשורת מכונים RFC (בקשות להערות – Request for Comments) וכל ספקי השירותים נצמדים אליהם מתוך בחירה.



הערה

מבנה האינטרנט דומה לזה של רשת טלפונים. אם תחשוב על כך, כאשר אתה מתקשר למישהו, השיחה עוברת מחברת הטלפון המקומית אל מפעיל שיחות חוץ, ומשם לחברת הטלפונים של האדם שאליו אתה מתקשר, ושם נוצר החיבור הסופי. יכול אף להיות מפעיל שירותי טלפון נוסף בדרך, כמו בשיחות בינלאומיות, למשל. כל חברות הטלפון המקומיות וחברות המפעילות שירותי שיחות חוץ מסכימות לחבר בין המערכות שלהן. אינך יודע בדיוק כיצד השיחה מגיעה אל היעד - אתה רק יודע שהיא הגיעה!

האינטרנט (Internet) היא רשת של **רשתות מחשבים** (WWW – World Wide Web) המאפשרת לכל אחד גישה. **רשת מחשבים** היא קבוצת מחשבים המחוברים בדרך המאפשרת להם "לדבר" זה עם זה. מחשב ברשת הנותן שירותי מידע הכוללים: טקסט, תמונות, סרטי וידאו, קטעי קול וכדומה, נקרא **שרת** (Server). מחשב ברשת המקבל שירות נקרא **לקוח** (Client). כל המחשבים "מדברים" בשפת תקשורת (פרוטוקול) אחידה, הנקראת **TCP/IP**.



פרוטוקול

כששני אנשים רוצים לדבר בינהם הם צריכים לדבר באותה שפה, בין אם זו שפה מדוברת (עברית, אנגלית, ספרדית) או שפת סימנים. גם ששני מכשירים אלקטרוניים רוצים "לדבר" אחד עם השני הם זקוקים "לשפה" משותפת. יש **שלט רחוק** (remote control) היודע "לדבר" עם הטלוויזיה, ויש שלט רחוק "שמדבר" עם שער הכניסה. נסה להפעיל את שער הכניסה בעזרת שלט הטלוויזיה ו... שום דבר לא יקרה. הסיבה היא ששני המכשירים (שלט הטלוויזיה ושער הכניסה) לא "מדברים" באותה "שפה". "שפה" זו נקראת **פרוטוקול** (protocol). "שפת" האינטרנט נקראת **TCP/IP**. זהו הפרוטוקול שבו כל המחשבים ברשת האינטרנט צריכים לדעת "לדבר". למעשה, TCP/IP הינו אוסף של פרוטוקולים לצרכי תעבורה שונים ברשת, כמו דואר אלקטרוני, העברת קבצים ועוד (נושא זה יפורט בהמשך).

שירותי האינטרנט

לאחר חיבורך לאינטרנט, יש לך גישה למיגוון שירותים. להלן רשימה חלקית של השירותים הבולטים:

- **גלישה** באמצעות דפדפן
- **דואר אלקטרוני** - E-mail
- **שרתי FTP** (File Transfer Protocol)
- **צ'ט** (Chat)
- **קבוצות דיון** (Newsgroups)

שירותים חדשים מוצעים כמעט מדי יום, ועם התפתחות הטכנולוגיה הם מאפשרים שימוש רב יותר באודיו (קול) וגם וידאו (סרטים). מספר השירותים רק ימשיך ויגדל.

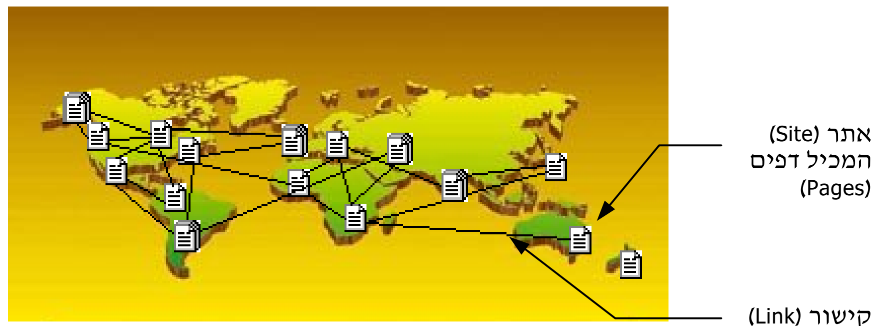
גלישה באמצעות דפדפן

World Wide Web, ובקצרה **WWW**, החל בשנת 1989 כאמצעי לפרסום מאמרי מחקר אקדמיים כדי שמדענים ברחבי העולם יוכלו לעיין בהם. מכיון שהתוכנה ששימשה הן לתצוגה והן לפרסום המידע הופצה לכל דורש, החלו משתמשים ברחבי העולם ליצור את מה שכונה מאוחר יותר בשם **אתרי Web** (Web sites). כיום, ה-Web הפך לאמצעי העיקרי לגישה למידע באינטרנט.

כיצד מאורגן המידע בשרת?

לאחר שהתחברת לאינטרנט (כמשתמש) תוכל לגשת למידע שנמצא במחשבי ה**שרת**. המידע במחשבי השרת מאורגן במבנה של **אתר** (Site). אתר מורכב מ**דפים** (Pages) המקושרים ביניהם, כך שבלחיצת עכבר והפעלת **קישור** (Link) ניתן לעבור מדף לדף. ניתן לעבור מדף לדף באותו אתר, או לעבור לדף אחר באתר אחר, במחשב שרת אחר.

לכן, באופן כללי ניתן לומר שרוב דפי המידע באינטרנט מקושרים ביניהם, זו המשמעות של מאגר ידע אנושי נגיש. הפעולה שבה עוברים מדף לדף נקראת **גלישה** (Surfing). כדי שניתן יהיה לגשת לאתר, יש לדעת את כתובתו Uniform Resource Locator (**URL**). זוהי כתובת ייחודית עולמית, כמו מספר טלפון. לפיכך, ייתכן שתמצא את אותו המספר (במקרה זה, שם החברה), אך הקידומת (ובמקרה זה הסיומת) תהיה שונה. לדוגמה, ייתכן שיהיה אתר שכתובתו www.coca-cola.co.il ואתר אחר ששמו www.coca-cola.com. הראשון הוא אתר מסחרי בישראל, ואילו השני הוא אתר בארה"ב.



מבנה ה-Web פשוט למדי. כאשר אתה רוצה לפרסם מידע שאנשים אחרים יוכלו לראות, הקבצים שלך צריכים להיות זמינים לאחרים באמצעות **שרת Web** (Web server). המסמכים, הנקראים גם **דפי Web** (Web pages), כתובים בעיקר ב-**HTML** (HyperText Markup Language) ויכולים לכלול מלל, גרפיקה, קול וגם וידאו. מדפי ה-Web שלך אתה יכול ליצור **hypertext links** (קישורים) אל דפים שנכתבו על ידי אנשים אחרים ונמצאים בשרתי Web אחרים. אתרים אחרים, במיוחד ספריות וכלי חיפוש, יוכלו ליצור גם הם **קישור** (link) אל האתר שלך, כדי שמשתמשים הפונים אליהם יוכלו למצוא את הדפים שלך. קישורים אלה בין אתרי Web יוצרים **סריג** (mesh), או **רשת עולמית** הדומה לרשת של קורי עכביש.

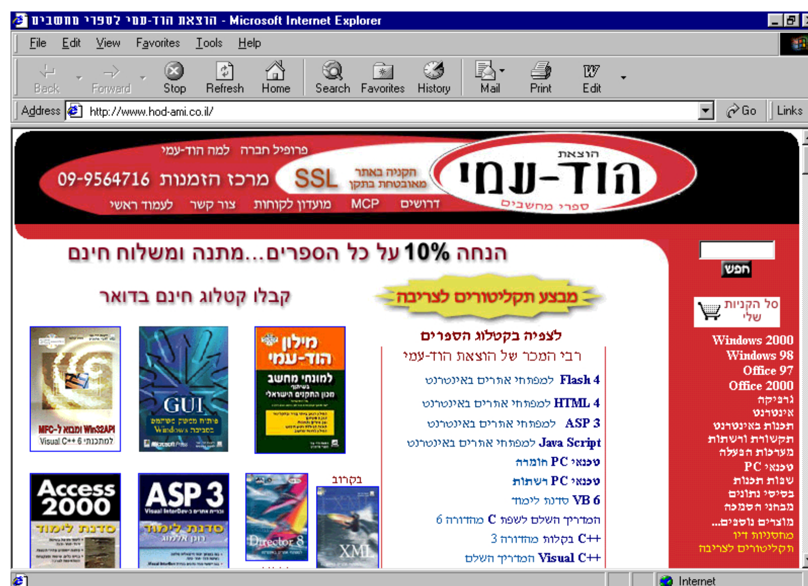


הערה

Microsoft כוללת שרת Web במערכת Windows NT/2000, כך שתוכל להתחיל לפרסם חומר באינטרנט מייד. ב-Windows NT/2000, שרת Web הוא IIS – Internet Information Server לעבודה מאומצת. במערכת הפעלה Windows 98 הוא נקרא PWS - Personal Web Server וב-Windows NT Workstation 4.0, שרת Web נקרא Peer Web Services ומיועד לשימוש קל. מעבר לשני שרתים אלה מבית Microsoft, קיימים כיום שרתי Web כמעט לכל סוג מערכת הפעלה/חומרה.

משתמשים מחפשים מידע ב-Web וגם מעיינים בו באמצעות תוכנת גלישה ברשת הקרויה **דפדפן** (browser). תוכנה זו מאפשרת למשתמשים לציין לאיזה שרת Web הם רוצים להתחבר.

הדפדפן הנפוץ ביותר הוא Internet Explorer של חברת Microsoft. הדפדפן מאפשר למשתמשים לראות גרפיקה ב**רזולוציה** (resolution) גבוהה, להשתמש בקול וגם וידאו, להציג דפי Web בעלי מבנה דף מורכב ולחפש בקלות אתרי Web חדשים.



ניתן להשתמש ב-Internet Explorer של Microsoft להצגת אתרי Web, כגון אתר הוצאת **הוד-עמי** בכתובת http://www.hod-ami.co.il.

	הערה Internet Explorer של Microsoft נכלל חינם בכל מערכת הפעלה של Microsoft. כמו כן, תוכל למצוא אותו בכל תקליטור של הוצאת **הוד-עמי** המצורף, אם מצורף, לספרי ההוצאה.

דפדפני הרשת מתקשרים עם שרתי Web השונים באמצעות **פרוטוקול HTTP** (HyperText Transfer Protocol). האינטראקציה (יחסי הגומלין) בין הדפדפן לבין השרת נעשת על ידי כך שהדפדפן שולח בקשה לקבלת מסמך מהשרת. השרת שולח את המסמך אל הדפדפן ובזאת (באופן פשטני ולא הכי מדוייק) מסתיים הקשר ביניהם. הקשר יפסק כאשר הדפדפן ישלח בקשה להפסיק את משלוח המסמך מהשרת (לחיצה על לחצן **עצור** (Stop)) או בגמר המשלוח מהשרת. הקשר יתחדש כאשר הדפדפן ישלח בקשה נוספת לקבלת מסמך מהשרת.

הצלחת Web והזמינות הגבוהה של דפדפני הרשת הפכה את טכנולוגיות Web לאמצעי פרסום עיקרי באינטרנט.

על Web	אם אתה רוצה ללמוד את כל הפיתוחים העדכניים בנושא אינטרנט, בקר בדף הבית של קונסורציום World Wide Web בכתובת http://www.w3.org/.

דואר אלקטרוני

דואר אלקטרוני (electronic mail) או **E-mail**, הוא הסיבה העיקרית שלשמה אנשים מתחברים (או מחברים ארגון) לאינטרנט. World Wide Web הוא האמצעי העיקרי שבו אנשים משתמשים לאיתור ולאחזור מידע, והדבר מרגש, מעניין ומסעיר ללא כל ספק. עם זאת, הדואר האלקטרוני הוא "סוס העבודה" של האינטרנט, השומר על קיום התקשורת.

כדי להשתמש בדואר אלקטרוני, המשתמש זקוק לתוכנת **לקוח** (client) של דואר אלקטרוני. תוכנה זו יכולה להיות ייעודית לדואר אינטרנט (למשל, Eudora), או שהיא יכולה להיות תוכנת דואר אלקטרוני שהיא חלק מתוכנה מקיפה, כמו Exchange Microsoft, Microsoft Mail או Lotus cc:Mail שיש לה **שער** (gateway) אל האינטרנט.

משתמשים שולחים הודעה זה לזה ומשתמשים בכתובות בסגנון אינטרנט, שנראות כך: user@domain. בכתובת זו, user הוא שם המשתמש המוקצה לו על ידי ספק השירותים (ISP), ו-domain הוא הכתובת עצמה כדוגמת hod-ami.co.il. שמות domains כוללים לרוב את שם החברה או שם הארגון, ולאחריו ציון ה-domain ברמה העליונה, בן שלושה תווים (ראה סעיף "The Domain Name System" בהמשך הפרק למידע נוסף אודות שמות תחומים, וההבדלים בין השמות בארה"ב ובארצות אחרות).

לדוגמה, כדי לתקשר עם הוצאת הוד-עמי, תוכל לשלוח דואר אלקטרוני לכתובת .info@hod-ami.co.il

הודעות דואר אלקטרוני נשלחות דרך האינטרנט, באמצעות רשת של **שרתי דואר** (mail servers) האחראים לאספקה וקבלה של דואר אלקטרוני. שרתים אלה משתמשים בעיקר ב**פרוטוקול SMTP** (Simple Mail Transfer Protocol) לשיגור וקבלת דואר אלקטרוני. דואר אלקטרוני המיועד למשתמש שהמחשב שלו **אינו מקוון** (offline) באותו רגע יכול להישמר בשרת, והמשתמש יוכל "למשוך" או לאחזר אותו מאוחר יותר באמצעות **פרוטוקול POP3** (Post Office Protocol 3).

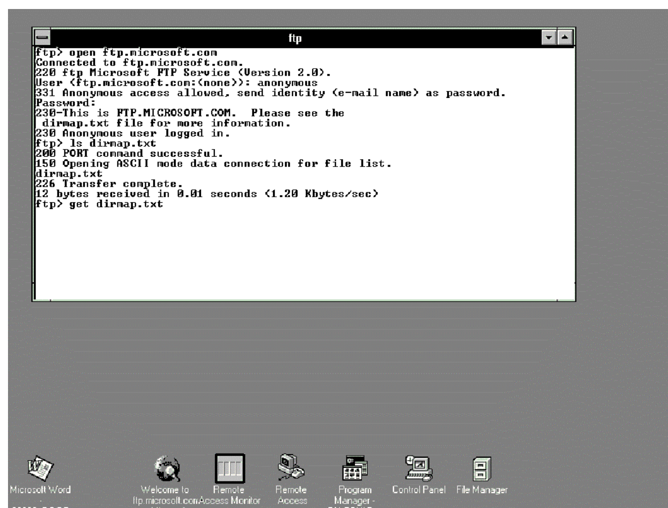
טיפ

נזכיר שוב, אל תתבלבל בעניין פרוטוקולי רשת. זכור ש-SMTP ו-POP3 הם פרוטוקולים למשלוח וקבלת דואר אלקטרוני באינטרנט, ואינם פרוטוקולי שידור ברשת.

שרתי FTP (File Transfer Protocol)

ב-Web תמצא כמויות גדולות של נתונים שרובם טקסט או גרפיקה. כשתרצה להוריד קבצי נתונים גדולים המכילים תוכנה או נתונים אחרים, תצטרך להשתמש ב**פרוטוקול העברת הקבצים - FTP** (File Transfer Protocol). בגלל השימוש הרב בו, פרוטוקול FTP נכלל כמעט תמיד בכל התקנה של מערכת הפרוטוקולים TCP/IP. מסיבה זו תראה שמרבית ספקי התוכנה מאפשרים גישה לתוכנות שלהם באמצעות שרת FTP. למשתמש המחובר לאינטרנט לא יהיה בהכרח דפדפן, אולם קרוב לוודאי שהותקנה אצלו תוכנת לקוח FTP.

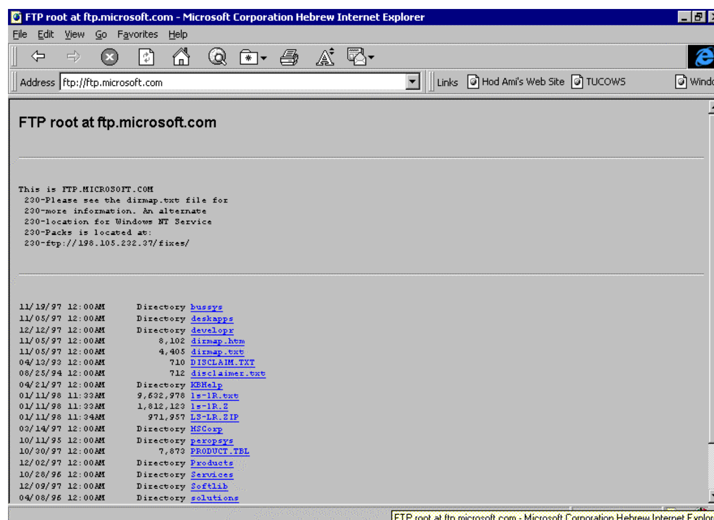
לקוחות FTP זמינים בשתי צורות: **מבוסס טקסט** (text based), ומבוסס **גרפיקה** (graphical). לקוחות FTP מבוססי טקסט החלו בעולם UNIX ולכן הם שומרים על מבנה הפקודות הקצרות ממערכת UNIX. מחלון טקסט, כמו שורת פקודה (ראה תרשים 14.2), ניתן לכתוב את הפקודה **ftp**, להתחבר למארח המפעיל שרת FTP ולהתחיל להוריד קבצים.



לקוח FTP מבוסס טקסט מסופק עם מערכות הפעלה רבות.

לקוחות FTP גרפיים זמינים בצורות רבות. קיימות תוכניות מיוחדות המיועדות לספק גישת FTP באמצעות ממשק גרפי. אולם לרוב תראה אנשים המשתמשים בדפדפני רשת להורדת קבצים. הן Internet Explorer של Microsoft והן Netscape Navigator, כמו גם דפדפנים אחרים, תומכים בהורדת קבצים באמצעות FTP, כפי שמוצג בתרשים הבא.

בגלל קלות אחזור הקבצים באמצעות הדפדפנים, משתמשים רבים ב-Web ימשיכו להוריד קבצים (כלומר, להעתיק אותם אל המחשב שלהם), בלי להיות מודעים כלל לכך שלעיתים הם משתמשים ב-FTP.



דפדפן Internet Explorer של Microsoft יכול לשמש להורדת קבצים באמצעות FTP.

Chat

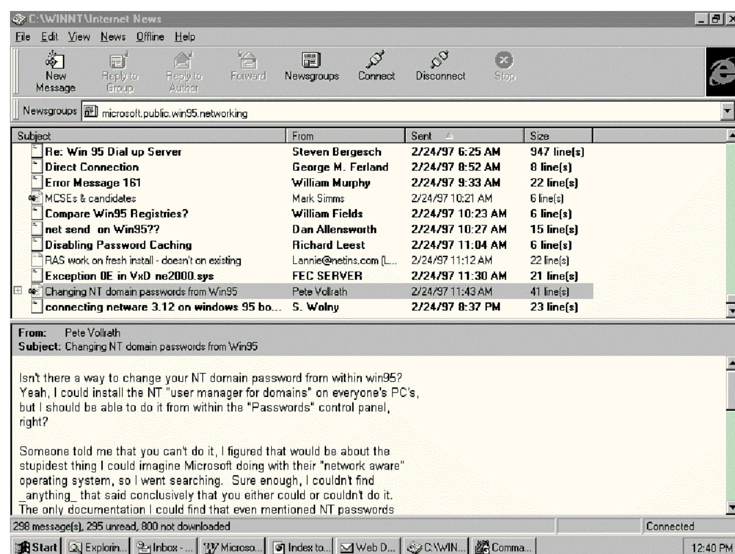
רבים ממשתמשי האינטרנט "מצ׳טצ׳טים" (מהמילה צ׳אט) ביניהם באמצעות תוכנות לקוח כגון ICQ או IRC. CHAT הוא קישור ישיר בזמן-אמת בין משתמשים המחוברים באותו זמן לאינטרנט. תוכנות CHAT רבות מאפשרות גם קישור וידאו וקול, במידה ורוחב הפס מאפשר זאת. למרות זאת, הקישור הטקסטואלי הוא עדיין הקישור הסטנדרטי.

קבוצות דיון (Newsgroups, Forums)

בתחילת התפתחות האינטרנט, נעשה מאמץ מקביל לפתח רשת המחברת בין מחשבים לקיום קבוצות דיון. רשת זו, המכונה UseNet, התפתחה לסדרת **קבוצות דיון** (newsgroups) ששותפו בין מחשבים באמצעות **פרוטוקול NNTP** (Network News Transfer Protocol) במשך הזמן, קבוצות דיון אלו החלו לעבוד דרך הדפדפן (פרוטוקול HTTP).

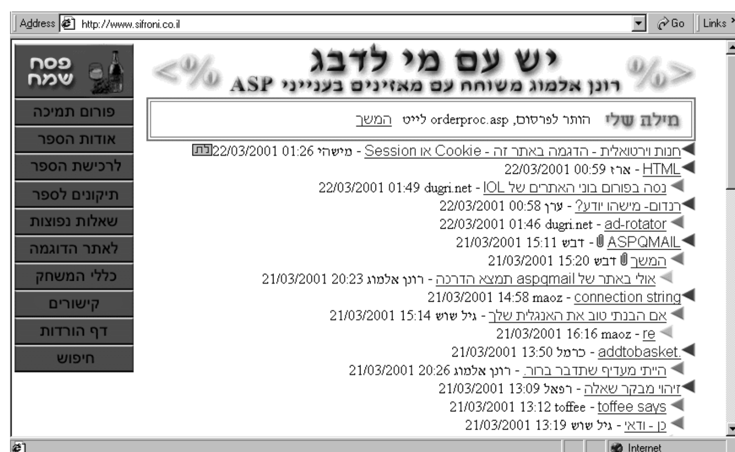
כיום, יש עשרות אלפי קבוצות דיון שונות הדנות כמעט בכל נושא שניתן להעלות על הדעת. ממערכות הפעלה לספורט, פוליטיקה, דת, בידור ועוד. אלפי מאמרים נשלחים לקבוצות דיון ברחבי העולם בכל דקה במשך היום.

כל המערכת פועלת באמצעות סדרה של **שרתי דיון** (news servers) המופעלים בדרך כלל על ידי ספקי שירותי אינטרנט, אוניברסיטאות או חברות. כל שרת דיון עוסק במספר קבוצות דיון המשותפות למשתמשים בשירותיו. כאשר אתה **שולח** (post) מאמר לקבוצת דיון, הוא זמין בשרת הדיון של ספק שירותי האינטרנט שלך. בתוך דקות עד שעות, המאמר יופץ לכל שרתי הדיון בעולם העוסקים בקבוצת דיון זו. בדרך זו מתרחש דיון כלל עולמי בנושא כלשהו.



ניתן להשתמש ב- Internet News של Microsoft לקריאת קבוצות דיון.

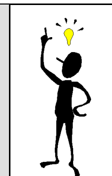
ניתן להשתתף בקבוצות דיון באמצעות תוכנית לקוח הנקראת **news reader**, כפי שמוצג בתרשים למעלה. אך, המגמה היא לאפשר ביצוע של כל הפעולות באינטרנט דרך הדפדפן. באתר www.sifroni.co.il תוכל לראות כיצד מתנהל פורום שכזה על ידי רונן אלמוג, מחבר הספר **ASP 3 סדנת לימוד**. הפורום מתנהל בדפדפן בעזרת טכנולוגיית ASP. בדרך זו הלקוח אינו צריך להתקין תוכנת לקוח מיוחדת לצורך השתתפות בקבוצת הדיון/פורום.



פורום המתנהל בסביבת הדפדפן.

Telnet

בימים הראשונים של האינטרנט, מחשבים מהירים היו יקרים ונדירים מאוד. למעשה, רשת ARPAnet המקורית נוצרה לשיתוף מחשבי-על בין חוקרים רבים. אחד הצרכים הנפוצים ביותר היה שחוקרים במקום אחד יתחברו למחשבי-על מרוחקים אלה, לביצוע מחקריהם עתירי החישובים. לצורך זה פותחה תוכנית Telnet. באמצעות הפקודה **telnet**, ניתן להתחבר למחשב מארח רחוק ולתקשר איתו כאילו היית מחובר אליו ישירות, כפי שמוצג בתרשים הבא. המארח המרוחק צריך להריץ Telnet server (הנקרא בדרך כלל Telnet daemon או Telnetd) ולאפשר למספר משתמשים להתקשר בו-זמנית.



הערה

Windows 95 ו-Windows NT מתקינים לקוח Telnet עם התקנת TCP/IP. קיים שרת Telnet ל-Windows NT Server כחלק מ-Windows NT 4.0 Resource Kit.

TCP/IP

מערכת פרוטוקולי TCP/IP נקראת במקרים רבים **מערכת פרוטוקול אינטרנט** (Internet suite protocol), מכיון שהיא פותחה במהלך פיתוח האינטרנט. לאחר מכן, שולב הפרוטוקול בכל שרתי UNIX ונכנס לשימוש נרחב ברשתות גדולות, הנקראות במקרים רבים **רשתות תאגיד** (enterprise networks). יכולת הגדילה של TCP/IP מרשתות קטנות לגדולות, והעניין הרב בחיבור רשתות מקומיות לאינטרנט אפשרו ל-TCP/IP להתפתח לפרוטוקול הרשת הנפוץ ביותר כיום.

חבילת פרוטוקולי TCP/IP כוללת מספר פרוטוקולים כמו (רשימה חלקית):

- **FTP/TFTP** (File Transfer Protocol). FTP מיועד להעברת קבצים בין מחשבים.
- **DHCP** (Dynamic Host Configuration Protocol) יכולת הקצאה של כתובת IP באופן דינמי.
- **SMTP** (Simple Mail Transport Protocol). SMTP מגדיר את התקן עבור שידור דואר אלקטרוני באינטרנט.
- **RIP** (Routing Information Protocol). פרוטוקול שבעזרתו הנתבים קובעים את הדרך הקצרה והפנויה ביותר להעברת המידע ברשת.

כתובות IP (IP Address)

ברשתות TCP/IP מוקצית לכל מחשב **כתובת IP** בת 32 סיביות, לדוגמה:

192.168.24.123

כתובת זו מחולקת לארבעה בתים בני שמונה סיביות כל אחד (octets), שכל אחד מהם נכתב כמספר עשרוני שערכו בין 0 ל-255, והם מופרדים זה מזה ב**נקודה** ("dot"). חלק מכתובת IP הוא **קוד זיהוי הרשת** (network id) אליה הוא שייך, ושאר חלקי הכתובת מציינים את **קוד זיהוי מארח** (host id) המציין את המחשב של המשתמש. לדוגמה, הצירוף 24.123 בכתובת שהוצגה קודם עשוי לזהות מחשב מסוים בתוך רשת TCP/IP שכתובתה 192.168.

בעת הפעלת הרשת המקומית שלך, באפשרותך לבחור כל תחום כתובות IP שתרצה. אולם, כאשר תחבר את הרשת לאינטרנט, ספק שירותי האינטרנט שלך (ISP) ימסור לך תחום כתובות IP שיהיה תקף עבורך.

כאמור, כל כתובת IP מחולקת ל-4 יחידות (classes) שהחלוקה הלוגית שלהן היא כתובת לרשת וכתובת לעמדה. קיימים חמישה סוגי Classes והם:

- **Class A**. כתובות אלו מיועדות לרשתות גדולות במיוחד, המשתמשות בבית (octet) הראשון לזיהוי הרשת ובשאר לכתובת העמדות. כתובות Class A היא בין 0 ל-126. כתובת 127 היא כתובת מיוחדת הנקראת loopback address.
- **Class B**. כתובות אלו מיועדות לרשתות בינוניות, המשתמשות בשני הבתים הראשונים לזיהוי הרשת, ובשני הבתים האחרים לכתובת העמדות. כתובות Class B בין 128 ל-191.
- **Class C**. כתובות אלו מיועדות לרשתות קטנות, המשתמשות בשלושת הבתים הראשונים לזיהוי הרשת ובבית אחד לכתובת העמדות. כתובות Class C בין 192 ל-223.
- **Class D**. בין 224 ל-239.
- **Class E**. בין 240 ל-247.

חומר נוסף על פרוקטולים באינטרנט תוכל למצוא בספרים הבאים בהוצאת **הוד-עמי**:

- **מבוא לתקשורת מחשבים**, ייעוץ מקצועי: נגה קרטס.
- **תקשורת מחשבים, פרוטוקולים וארכיטקטורות רשת**, גולן מוגרבי.
- **המדריך השלם לטכנאי PC - רשתות תקשורת**, דורון סיון.

Subnet masks

כשאתה מתכונן להתקשר למישהו בטלפון, כיצד אתה יודע אם צריך לחייג תחילה את קידומת אזור החיוג, או שניתן לחייג את המספר המקומי בלבד? בארה"ב התשובה נמצאת בשימוש בקידומת אזור חיוג בת 3 ספרות, ובארץ - שתי ספרות. בהנחה שאתה יודע את אזור החיוג שלך ואתה מקבל מספר טלפון בעל אזור חיוג התואם לשלך, אתה יודע שאין צורך לחייג את הקידומת. אולם כשנמסר לך מספר טלפון בעל קידומת שונה, עליך לחייג תמיד את קידומת אזור החיוג כדי להגיע למספר טלפון זה.

מחשבים ברשתות TCP/IP פועלים בצורה דומה. אולם, בעוד שאתה יודע ששלוש הספרות הראשונות של מספר טלפון הן קידומת אזור החיוג, מחשבים אינם יודעים מייד כמה מתוך 32 הסיביות בכתובת IP מייצגות את ה**רשת** (Network ID) וכמה מהן מייצגות את **זיהוי המארח** (Host ID) ברשת. כדי להבחין בין זיהוי הרשת לבין זיהוי המארח, משתמשים המחשבים ב-Subnet mask האומרת להם כיצד לקרוא את הכתובת.

	רעיון מפתח בעת משלוח מנת נתונים אל מחשב אחר ברשת TCP/IP, המחשב משתמש ב-Subnet Mask כדי לקבוע את חלקי הרשת והמארח בכתובת היעד.

ב-Subnet Mask המספר 255 מציין את כתובת הרשת, והמספר 0 מציין את כתובת העמדה. דוגמה ל-Subnet Mask : 255.255.0.0 המתאים לכתובת מ-Class B.

Dynamic Host Configuration Protocol - DHCP

כאשר רשתות גדולות החלו להשתמש ב-TCP/IP, תהליך הקצאה ידנית של כתובות IP לכל מחשב הפך למסורבל. מעקב אחר כתובות IP שכבר הוקצו הסתבך, כאשר מחשבים הועברו ממקום למקום בארגון. כדי לפשט את משימת הקצאת כתובות IP, פותח פרוטוקול **DHCP** (Dynamic Host Configuration Protocol) המשמש להקצאת כתובת IP דינמית.

כדי להשתמש ב-DHCP, מפעילים שרת DHCP במקום כלשהו ברשת, ומספקים לו תחום כתובות IP שמתוכן אפשרי להקצות באופן דינמי כתובת לכל מחשב שמבקש. לאחר מכן, צריך לעבור בין כל מחשבי הלקוח ברשת ולהגדיר להם בקשת כתובת IP משרת DHCP. כאשר מחשבים אלה מאותחלים לראשונה (ובכל פעם שיופעלו), הם משדרים **הודעה** (broadcast message) ברשת המקומית המבקשת כתובת IP משרת DHCP. שרת זה מקצה כתובת לכל מחשב המבקש זאת, עד שאוזלות לו כל כתובות IP.

הערה

למעשה, כל כתובת IP **חכורה** (leased) למחשב הלקוח לפרק זמן נתון, כל עוד הוא מקיים את הקשר. ניתן לחדש חכירת כתובת בכל עת שהמחשב זקוק לה.

אחד היתרונות של DHCP, הוא בכך שניתן להעביר מחשבים בחברה ממקום למקום ללא דאגה לכתובת IP שלהם. כאשר מחשב מחובר ל**מקטע** (segment) רשת אחר ומופעל, הוא משדר הודעה במקטע אליו הוא שייך ומקבל כתובת IP משרת DHCP במקטע הרשת החדש שלו. תהליך זה מאפשר למנהלי רשתות להקצות במרוכז תחומי כתובות IP לשרתי DHCP ולעקוב בקלות רבה יותר אחר הכתובות המוקצות.

רעיון מפתח

DHCP מספק מנגנון להקצאה דינמית של כתובות IP.

The Domain Name System

TCP/IP הוא פרוטוקול התקשורת באינטרנט. מחשבים מתקשרים באמצעות כתובות IP בעלות מבנה מספרי, כגון 192.168.10.123. למרות ששיטה זו מתאימה למחשבים, לאנשים יש קושי רב לזכור את כל המספרים האלה ודוגמתם. אם מישהו היה אומר לך "לקבלת מידע נוסף אודות החברה שלנו, בקר באתר Web בכתובת http://207.22.13.4", קרוב לוודאי שהיית מחייך אליו בנימוס ושוכח את הכתובת.

כדי להתמודד עם הצורך לזכור כתובות, פיתחה קהילת האינטרנט את Domain Name System, המוכרת בשם **DNS**. במערכת זו, מוקצה לכל ארגון domain name, כגון **microsoft.com** או productivitypoint.com, המזהה באופן ייחודי את הארגון ברחבי האינטרנט.

בכל פעם שאתה מתקשר ברשת לכתובת, כגון **www.hod-ami.co.il**, המחשב שלך מציג שאילתה **לשרת DNS** (DNS server) מקומי (הנקרא גם name server) כדי לברר את כתובת IP של www.hod-ami.co.il. כאשר שרת DNS מחזיר את כתובת IP המתאימה, המחשב שלך שולח את מנות הנתונים אל כתובת זו. כל התקשורת בין מחשבים באינטרנט מבוססת על שימוש בכתובות IP. אם מסיבה כלשהי שרת DNS שלך אינו יכול למצוא את כתובת IP עבור שם ה-domain שהקלדת, לא תיווצר תקשורת ותקבל הודעה על כך.



הערה

DNS משתמשת בהיררכיה שלמה של שרתי DNS מבוזרים לפענוח כתובות. אם שרת DNS המקומי יודע את כתובת IP של www.microsoft.com, הוא שולח אותה חזרה למחשב. אם לא, הוא מעביר את הבקשה אל שרתי DNS ברמה גבוהה יותר, כדי לדעת איזה שרת DNS אחראי על microsoft.com. כלומר, שרת DNS המקומי מתקשר עם שרת DNS מרוחק זה כדי לברר את כתובת IP של www.microsoft.com, מחזיר את כתובת IP אל המחשב שלך ומאחסן אותה במחשב המקומי למקרה שאתה (או אחרים ברשת המקומית) תרצה לבקר באתר זה שוב.

Domain names מחולקים למספר תחומים ב**רמה עליונה** (top-level domains), כפי שמוצג בטבלה הבאה. לארגונים וחברות שנמצאים מחוץ למדינות ארה״ב, יש בדרך כלל קוד נוסף לציון המדינה. קוד המדינה הוא בן שתי אותיות.

DNS Top-Level Domains (רשימה חלקית)

תחום	**תיאור**
.com	(commercial) ארגונים מסחריים.
.edu	(education) מוסדות חינוך, כגון מכללות ואוניברסיטאות.
.gov	(government) סוכנויות ממשל, בעיקר הממשל הפדרלי של ארה״ב.
.mil	(military) ארגונים צבאיים ואתרים צבאיים של ארה״ב.
.net	(network) ספקי שירותי אינטרנט וארגונים אחרים הקשורים לרשתות (שים לב שספקי שירותים אחדים משתמשים גם ב-com.).
.org	(organization) ארגונים שלא למטרת רווח, וארגונים אחרים שאינם מתאימים לקטגוריות אחרות.
.int	(international) ארגונים בינלאומיים.
.k12	פרוייקטים חינוכיים.
.xx	קודי מדינה בני שתי אותיות המציינים את מיקום הארגון (לדוגמה, il. עבור ישראל, fr. עבור צרפת, au. עבור אוסטרליה).

הקצאת שמות domains עבור ארה״ב ורוב מדינות העולם מטופלת כיום על ידי הארגון InterNIC. הארגון מקצה גם כתובות IP, והוא מופעל על ידי חברה פרטית על פי חוזה עם ממשל ארה״ב. הדבר ישתנה ככל הנראה בשנים הקרובות, מכיון שקיים מאמץ להוסיף TLD נוספים ולאפשר לארגונים אחרים מלבד InterNIC לרשום שמות domains.

למעשה, כדי לקבל שם domain, אתה קובע איזה domain ברמה העליונה מתאים לארגון שלך, ואז מנסה למצוא שם שאינו בשימוש על ידי מישהו אחר. זהו המאבק האמיתי של עולם DNS, אשר גרר אין ספור תביעות משפטיות הקשורות בסימנים

מסחריים, זכויות יוצרים וכדומה. מכיון שיכול להיות שם ייחודי **אחד** בשם domain, ארגונים בתחומי תעשייה שונים עם שמות או ראשי תיבות דומים, אינם יכולים להשתמש באותו domain name.

לאחר קבלת Domain name, אתה או ספק השירותים שלך חייבים להפעיל שרת DNS שיענה על בקשות מאתרי אינטרנט אחרים לקבלת מידע אודות ה-domain שלך. אתה אחראי להקמת שמות, כגון www.yourcompany.domain, וקישור או הצבעה שלהם אל כתובות IP תקפות.



רעיון מפתח

DNS (Domain Name System) היא מערכת המתרגמת domain names לכתובות IP, ומאפשרת לאנשים להתקשר עם שירותי אינטרנט כמו ה-Web, באמצעות שמות טקסט במקום מספרים.

קבלת תחום כתובות IP תקף מספק השירותים

כאשר אתה משתמש במערכת פרוטוקול TCP/IP ברשת הפנימית שלך (Lan), אתה יכול להשתמש בכל תחום של כתובות IP. אך, ברגע שתחבר את הרשת לאינטרנט, עליך להשתמש בתחום כתובות IP ייחודי שאף אחד אחר בכל רחבי האינטרנט אינו משתמש בו.

חשוב על כך במונחי מערכת הטלפון שבה כל מספר טלפון הינו ייחודי. כאשר אתה מחייג בארה״ב 603-471-0848, אתה מגיע לטלפון מסוים בארה״ב, אך כשאתה מחוץ לארה״ב עליך להוסיף למספר זה את קידומת המדינה. שילוב קידומות מדינות ומספרי טלפון ייחודי במדינה מספק לך ״כתובת״ ייחודית לכל מספר טלפון בעולם. אם המערכת לא היתה פועלת כך, השיחות לא היו עוברות.

כך גם בעולם האינטרנט, לכל מחשב צריכה להיות **כתובת IP ייחודית**. בדומה למספר טלפון, כתובת זו תזהה את המחשב שלך בפני שאר העולם בעת שתשתמש באינטרנט.

לאחר בחירת ספק שירותים, עליך לקבל ממנו תחום כתובות IP עבור המחשבים ברשת שלך. רוב ספקי השירותים קיבלו הקצאה של תחום כתובות מ-InterNIC, הארגון האחראי היום על הקצאת כתובות IP ושמות תחומים. כאשר תקבל את תחום הכתובות, עליך להבטיח שכל המחשבים בארגון שלך שייגשו לאינטרנט ישתמשו בפרוטוקול TCP/IP עם כתובות חדשות אלו. אם לא תשתמש בכתובות תקפות, המחשבים שלך לא יוכלו לתקשר עם אתרי אינטרנט אחרים.

ספק השירותים גם יכול לפעול יחד איתך לקבלת domain name תקף עבור הארגון שלך. שם זה, כמו למשל microsoft.com, יזהה אותך בפני העולם הן עבור כתובות דואר אלקטרוני והן עבור מאמצי הפרסום באינטרנט (אתרי Web, שרתי FTP וכדומה). כפי שהוזכר בסעיף הקודם, מישהו (אתה או ספק השירותים שלך) יצטרך לספק שירות שם DNS עבור domain name זה.

סיכום

רשת האינטרנט של היום, שהחלה במחקר צבאי בתקופת המלחמה הקרה, הפכה לאמצעי תקשורת עולמי. למעשה, זוהי רשת תקשורת מרחבית ענקית בפרוטוקול TCP/IP המורכבת מרשתות רבות של ספקי שירותי אינטרנט המחוברות ביניהן. לאחר התחברות לאינטרנט, יש לך גישה למספר גדול של שירותים ובהם World Wide Web, דואר אלקטרוני, FTP וקבוצות דיון.

כל התקשורת באינטרנט מתרחשת בין מחשבים המשתמשים בכתובות **IP**. אולם, כדי לסייע למשתמשים לזכור כתובות פותחה - **DNS** (Domain Name System). DNS מאפשרת לנו לקשר domain name עם כתובת IP ולהשתמש בשמות בעלי משמעות לצרכי התקשרות.

כדי להתחבר לאינטרנט, עליך לדאוג לחיבור באמצעות **ספק שירותי אינטרנט - ISP** (Internet Service Provider). קיימים סוגים רבים של ספקי שירותים, והבחירה עבור הארגון שלך חשובה מאוד. לאחר בחירת ISP, עליך לקבל תחום כתובות IP תקף ו-domain name.

חלק 2 - HTML

פרק 1 - היכרות עם HTML

פרק 2 - תגיות

פרק 3 - טקסט ועיצוב

פרק 4 - קישור

פרק 5 - תמונות

פרק 6 - צבע

פרק 7 - טבלאות

פרק 8 - טופס

פרק 9 - HTML בעברית זה LMTH

פרק 10 - מסגרות

פרק 1

היכרות עם HTML

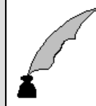
אם החלטת שאתה רוצה אתר באינטרנט, הגעת למקום הנכון!

הבה נראה איך כל העניין הזה עובד...

קצת על HTML

נוכל לספר הרבה על תהליך התפתחות רשת ה-WWW ורשתות בכלל ועל התפתחות שפת HTML בפרט. אך נושא זה כבר נדוש ומוכר. לכן, נפרט רק במספר שורות בהמשך הפרק על ההיסטוריה של HTML ושל ה-Web.

שפת HTML היא הבסיס לפיתוח אתרים באינטרנט.



בעברית פשוטה!

HTML הם ראשי תיבות של HyperText Markup Language. תבין בוודאי שזהו הבסיס לאתרי אינטרנט, עליו נבנים שאר הדברים.

HTML זו לא שפת תכנות

ראשית, מספר חוקי יסוד. החוק הראשון אוסר על השימוש במילה "תכנות" בעבודה עם HTML, מכיון ששפת HTML אינה שפת תכנות. HTML היא שפה שבעזרתה אנו קובעים את מראה המסמכים המוצגים בעזרת תוכנה שנקראת "דפדפן". למעשה, לא מתבצע כאן שום תהליך לוגי ולא מתבצע הידור של הקוד, בניגוד לשפות תכנות. כל מי שהתנסה אי-פעם בתכנות, ודאי יבין במהירות ששפת HTML אינה מתקרבת ביכולת שלה לשפות תכנות, ושהיא פשוטה הרבה יותר לשימוש, וקלה מאוד ללמידה.

השם **HyperText Markup Language** (HTML) מסגיר את היותה **שפת כתיבה** (Markup Language) או **שפת סימון**. שפת כתיבה היא שפה המגדירה כיצד ייראה עמוד טקסט מסוים. לדוגמה, העיצובים השונים שאנו נותנים לטקסט במעבד תמלילים מתבצעים ברקע כשפת סימון. למעשה, כשאנו כותבים מסמך אינטרנט, אנו מבצעים כמעט את אותן פעולות שאנו מבצעים במעבד תמלילים, כגון הגדלה והקטנה של טקסט, הדגשה, קו תחתון והחלת סגנונות כותרת בגדלים שונים.

למה דווקא HTML?

אז מה כל כך מיוחד ב-HTML? לפי מה שראינו, אולי פשוט יותר לשלוח את מסמכי ה-Word המוכרים לנו על גבי רשתות תקשורת ולהקל על החיים שלנו?! ובכן, כפי שתראה מייד, לשפת HTML יש מספר יתרונות על מעבד התמלילים, שגורמים לה להיות שפת כתיבה מועדפת מאשר מסמכי מעבד התמלילים שלך:

- HTML תוכננה לעבודה ברשת, בעוד שכל מעבדי התמלילים הקיימים כיום בשוק מתוכננים לעבודה על סוג מחשב מסוים, ומקפידים מאוד לגבי סוגי הגופנים בהם נעשה שימוש. HTML תוכננה לעבוד על כל מחשב, מכל סוג, בכל מקום והיא סלחנית מאוד לגבי פרטים קטנים.
- HTML היא תקן בינלאומי. בהמשך, נדבר על התקנים של HTML בהרחבה, אבל בינתיים חשוב להבין ש-HTML אינה שייכת לחברה מסוימת, בכל אופן לא באופן ש-Word שייכת לחברת Microsoft. HTML מותרת לשימוש על ידי כל מפתח אתרים, ללא צורך ברכישת זכויות יוצרים מחברה כלשהי, והיא ניתנת לקריאה בכל סוגי המערכות.
- HTML קריאה על ידי אנשים. למעשה, זהו חלק הארי של העיצוב. האם קרה לך פעם שפתחת מסמך WORD בעורך טקסט פשוט (למשל, בפנקס הרשימות של Windows)? זה לא מראה מלבב במיוחד. שפת HTML נולדה כדי שתהיה מובנת לכל אחד, לא רק לתוכניתנים.
- HTML מיישמת קישורים (גם מעבד תמלילים Word 97/2000). אם בילית בשיטוט ברחבי האינטרנט, אתה ודאי מכיר את אותן המילים המסומנות בצבע וקו תחתון, בעזרתן אתה יכול לעבור ממקום למקום. ובכן, HTML מתוכננת כך, שלנו המפתחים יהיה קל מאוד להוסיף קישורים למסמכי HTML נוספים ו/או לסוגים שונים של נתונים. בזה כוחה הגדול של השפה.
- HTML תומכת במולטימדיה. זהו הנימוק המשכנע ביותר מפתחים להשתמש בשפה זו ליצירת דפי Web לאינטרנט. אם יש משהו שמפריד את האינטרנט, כמדיה למידע ופרסום, ממדיות אחרות (כגון: עיתונות ורדיו), זו היכולת להציג נתונים בעזרת טקסט, צליל, תמונה, וידאו ועוד כמו בטלוויזיה...

אלו הן, על קצה המזלג, היכולות של HTML. כעת, הבה נביט מקרוב על השילוב בין קישור לבין מולטימדיה, מושג הידוע בקרב מפתחי HTML כ**קישור**.

קישור

HyperText הוא טקסט המסומן בצורה מיוחדת כדי לאפשר קישור בין מסמכים ובתוכם ברשת. בדרך כלל, ה**קישור** מופיע כטקסט המסומן בקו תחתון, אבל קישור יכול להיות גם תמונה. כאשר אתה מציב את סמן העכבר על קישור הוא משנה את צורתו ל- . לחיצה אחת על העכבר יכולה להעביר אותך למסמך אליו מקושר הטקסט המסומן, ולגרום לו להופיע במסך הדפדפן שלו.

כך הופך הקישור את מסמך האינטרנט לפשוט וקל לשימוש ולדפדוף, ובונה את הבסיס למבנה הסדיר של רשת מידע. רשת מסמכים המקושרים ביניהם מאפשרת מעבר אינטואיטיבי בין מסמכים ונושאים.

מתוך היכרות עם הצורה בה פועלת רשת האינטרנט, תוכל לתאר לעצמך אתר בו קיים קישור המתייחס לקבוצת הכדורסל ״מכבי תל-אביב״. אתה לוחץ על המילה המסומנת, סקרן להיכן תגיע! זה יכול להיות אתר החדשות של ערוץ הכבלים, המזכיר את הקבוצה במילים ספורות בלבד, או לחילופין לאתר הקבוצה הרשמי, בו מופיע מידע מפורט לגבי כל מרכיבי הקבוצה, מהאוהדים, דרך השחקנים ועד המנקה.

באתר ״מכבי תל אביב״, אתה נתקל בטקסט שמדבר על אירוע כלשהו. אתה לוחץ לחיצה קלה, בעזרת העכבר, ומגלה להפתעתך שהגעת לאתר של חברת כרטיסים המוכרת כרטיסים לאירוע. בין שאר המידע המופיע באותיות קטנות, תמצא גם מידע על אפשרות נסיעה לאירוע זה אבל גם לאירועים אחרים שאינם קשורים בספורט כמו: מופע של ריטה, קונצרט בניצוחו של זובין מהטה, פסטיבל הזמר החסידי בצפת, ושאר האתר העמוס במידע על אירועים ברחבי העולם. מכאן תוכל, כמובן, להגיע לאתרים שונים ומשונים ובנושאים מגוונים, וכל זאת בעזרת לחיצות אינטואיטיביות על לחצן העכבר.

אז איפה התחלנו ואיפה סיימנו? זה הקסם של הקישור. באפשרותנו לקבוע לאן אנו גולשים ברשת האינטרנט. בחירה זו היא מקסימה, אך בעיקר מהווה את עמוד התווך עליו בנויה הרשת.

HTML וגרסאותיה

״האבא״ של HTML הוא תקן הנקרא **SGML** המיועד להעברת מסמכים אלקטרוניים וניהול מסמכים. זהו תקן בעל עושר תחבירי וגמישות, אבל זו היתה ונשארה בעייתו הגדולה ביותר - רק מתכנת בכיר בעל רקע וניסיון יכול לעצב מסמך שיעמוד בחוקי התקן. לקחו מ-SGML מספר הגדרות ותגיות, הגדירו מספר מבנים פשוטים ויצאו לדרך. וראה זה פלא - תקן HTML התפשט כמו שריפה בשדה קוצים.

בוודאי שמת לב לכך, שבהקשר למילה HTML תמיד נדחף עוד מספר מעצבן כלשהו שאמור להגיד לנו משהו לגבי HTML: HTML 2, HTML 3, HTML 3.2, HTML 4. מספרים אלה מייצגים את גרסת HTML, הנקבעת על ידי ועדה העוסקת בנושא התקינה.

בוועדת התקינה חברים נציגים של החברות הגדולות העוסקות בפיתוח לאינטרנט, כמו : Microsoft, Netscape, Sun ו-Apple.

התקינה בתחום זה היא בידי ארגון עולמי הנקרא World Wide Web Consortium, או בקיצור W3C בכתובת **www.w3c.org**. בארגון זה חברים נציגי החברות המוזכרות ונציגים מחברות רבות נוספות. אחת השאיפות העיקריות של הארגון היא להגיע למצב שכל משתמש ברשת יוכל ליהנות מחלקם הגדול של האתרים, וממה שיש להם להציע, ללא התחשבות בדפדפן בו הוא משתמש.

HTML 4 הינו התקן האחרון, שהוכרז ב-18 בדצמבר 1997.

XHTML 1.0 הינו תקן הנמצא בדרך בין HTML 4 לבין XML אבל הרבה יותר קרוב ל-HTML 4 מכל תקן אחר. התקן הוכרז בינואר 2000. באופן כללי ניתן לומר שהשינויים בין HTML 4 ל-XHTML הם שינויים קוסמטיים בלבד ברמת קוד המקור ואין להם השפעה על מראה הדף.

HTML, זה הכל?

לא. כתיבת דפי Web בעזרת HTML היא הבסיס ליצירת אתר באינטרנט. מעבר לכך קיימים כלים נוספים שמשתלבים יחד, והם :

JavaScript - שפת תכנות הנכתבת במסגרת דף HTML שבעזרתה ניתן לנהל דו-שיח עם המשתמש בצורה הרבה יותר טובה מאשר עם קישורים בלבד. שילוב שפת תכנות זו מאפשר יצירת אינטראקטיביות בין האתר למשתמש בו. להוספת JavaScript אינך צריך כל תוכנה אחרת, עורך הטקסט שברשותך יספיק.

הנה כמה מהדברים שניתן לעשות עם JavaScript :

- **בניית טופס**. כאשר המשתמש ילחץ על לחצן **שלח טופס**, תוכל לבדוק מה רשם בו המשתמש ולהעיר לו בהתאם, אם לא מילא שדה בטופס או מילא ערך שגוי. תוכל גם להציג בפניו שוב את הטופס, כדי שיתקן את הטעון תיקון.
- **מקום הימצאות הסמן**. תוכל לדעת היכן נמצא הסמן, ובהתאם לכך להציג תכנים מבלי שהמשתמש לחץ בעכבר. מספיק שהוא העביר את הסמן מעל מקום שקבעת. למשל, כאשר המשתמש מעביר את הסמן מעל טקסט מסוים, רקע הטקסט משתנה. למשל, כאשר המשתמש מעביר את הסמן מעל תמונה מסוימת, היא מתחלפת בתמונה אחרת.
- **עבודה עם חלונות**. הפעלת קישור בדף HTML מציגה תוכן של דף HTML באותו חלון או בחלון חדש. עם JavaScript תוכל לשלוט על גודל החלון, כמו גם על הלחצנים ושורת הכותרת שלו.

לימוד שפת **JavaScript** בחלק השלישי של ספר זה.

ספרים נוספים בהוצאת **הוד-עמי**:

JavaScript למפתחי אתרים באינטרנט

JavaScript המדריך השלם

Java - שפת תכנות המיועדת לבניית **יישומונים** (Applets), במיוחד עבור אתרי אינטרנט. לצורך כתיבת יישומון Java יש צורך בערכת פיתוח תוכנה, שאינה כלולה בכלים שהוזכרו עד כה, אותה ניתן להוריד בחינם מהאינטרנט. שפת Java הינה שפת תכנות, כמו למשל ++C. תוכל בעזרתה לכתוב כל תוכנית שתעלה בדעתך, החל מתוכנית המקבלת מספר מהמשתמש וכופלת אותו ב-2, דרך בניית טפסים לקליטת נתונים, משחק מחשב ועד תוכנית לניהול חשבון בנק. אם אתה מעוניין, מסיבה כלשהי, לכתוב תוכנית לאינטרנט שתוכל לפעול, מבלי שלמשתמש תהיה יכולת להסתכל בקוד - השתמש ב- Java.

לימוד **Java** לא נכלל במסגרת ספר זה.

ספרים ללימוד Java בהוצאת **הוד-עמי**:

Java 2 למפתחי אתרים באינטרנט

The Java Tutorial סדנת לימוד

DHTML - המשך המגמה של יצירת דפי HTML אינטראקטיביים ודינמיים ככל האפשר. שימוש ברכיבי מערכת מובנים (חלק ממערכת Windows) ורתימה שלהם לדפי HTML. נחמד ויפה, אבל מכיון שזוהי טכנולוגיה של Microsoft, משתמשי Netscape לא יראו דבר, כלום!

לימוד **DHTML** אינו נכלל בספר זה.

ספרים ללימוד **DHTML** בהוצאת **הוד-עמי**:

HTML 4 למפתחי אתרים באינטרנט

VBScript - שפת תסריט, דומה בגישתה ל-JavaScript אבל היא של Microsoft. כלומר, מי שמשתמש בדפדפן Netscape לא יוכל ליהנות מטכנולוגיה זו. לכן, אם אתה מתכוון להשתמש בשפת תסריט במסגרת דפי HTML שהינך כותב, עשה זאת עם JavaScript.

לימוד **VBScript** אינו נכלל בספר זה.

היכרות עם **VBScript** תוכל לעשות בעזרת הספר:

HTML 4 למפתחי אתרים באינטרנט

ASP - טכנולוגיה נוספת מבית Microsoft שעובדת רק על שרתי Web מסוג **IIS** (תוכנת שרת מבית Microsoft) המאפשרת, בעיקר, ניהול מסדי נתונים באינטרנט, החל ממועדון לקוחות, דרך חנות ועד ניהול חשבונות בנק. ASP זה המנוע שמאחורי הטופס והלחצן שמבצע פעולות ברקע, בעיקר של טיפול בנתונים: הוספה, מחיקה, עדכון, יצירת דוח או ביצוע שאילתה.

לימוד **ASP** בחלק הרביעי של ספר זה.

ספרים נוספים ללימוד **ASP** בהוצאת **הוד-עמי**:

ASP 3 ו-Visual InterDev סדנת לימוד

ASP 3 המדריך השלם

ASP 3 למפתחי אתרים באינטרנט

Flash - בניסיון להעלות עוד את הרף בכל הקשור למולטימדיה באינטרנט, הופיעה תוכנת Flash. בעזרתה יוצרים הנפשה, הכוללת פעולות אינטראקטיביות וטפסים המיועדים לאינטרנט. תוכנה זו עשירה בתנועה ובצליל ונטענת במהירות. סרטי Flash בונים בעזרת תוכנת Flash אותה יש לרכוש.

לימוד **Flash** אינו נכלל במסגרת ספר זה.

ספרים ללימוד **Flash** בהוצאת **הוד-עמי**:

Flash 5 כולל Action Scripts למפתחי אתרים באינטרנט

Flash 4 למפתחי אתרים באינטרנט

Director – תוכנה ליצירת סרטי אנימציה, סרטונים אינטראקטיביים, מצגות עסקיות, יישומי מסחר אלקטרוני ועוד להפצה על גבי תקליטורים וגם באינטרנט.

לימוד **Director** אינו נכלל במסגרת ספר זה.

ספרים ללימוד **Director** בהוצאת **הוד-עמי**:

Director 8 למפתחי אתרים באינטרנט

XML – תקן שמקורו בשפת SGML, בדומה לשפת HTML. בשפת HTML הדגש הוא על עיצוב: גודל גופן, צבע רקע וכדומה. ב-XML הדגש הוא תוכן. זה לא אומר שה-HTML "מת", אלא רק נוספו לו יכולות של שפת תכנות ומסדי נתונים.

לימוד **XML** אינו נכלל במסגרת ספר זה.

ספרים ללימוד **XML** בהוצאת **הוד-עמי**:

XML למפתחי אתרים באינטרנט

יצירת מסמך HTML ראשון

תאמין או לא, עוד מעט תתחיל ליצור דפי HTML. כדי שלא תיכנס לפינות חשוכות, נעבור בקצרה על המרכיבים הבסיסיים ביצירת דפי HTML, איזה כלים נחוצים לך וכיצד תשמור אותם.

כדי לפתח דפי HTML, תזדקק לעורך טקסט פשוט. תוכל להשתמש ב**פנקס הרשימות** של Windows (Notepad).

בנוסף לעורך טקסט, נחוץ לך דפדפן שבעזרתו תוכל לבדוק את תוצאות יצירתך. רצוי שיהיה ברשותך Microsoft Internet Explorer, כדי שתוכל לראות את מה שיצרת ובעתיד לאפשר (אם תרצה, כמובן) לגולשים ברשת ליהנות מהם.

	הערה הסיבה להעדפת השימוש ב-IE היא התמיכה של הדפדפן בעברית.

כעת, הגיע הזמן ללמוד כיצד ליצור דפי HTML. נביט ב**תגיות** (HTML Tags) בהן נשתמש לבניית אתר. לאחר שתבין את המושג **תגיות**, ניגש לצעד הבסיסי ביותר בלימוד HTML: התגיות בהן תשתמש בכל דף HTML שתכתוב. תוך כדי עבודה תיווכח שאנו יוצרים תבנית בסיסית, עליה נוכל לבסס כל סוג דף באתר אינטרנט.

בפיתוח באינטרנט תיתקל פעמים רבות במונח **תגיות HTML**. **תגיות** הן פקודות בסיסיות שעליהן בנויה הכתיבה ב-HTML. כמעט כל דבר בדפי אינטרנט מושתת עליהן.

תגיות תופענה בקוד HTML תמיד בתוך סוגריים זוויתיים: <תגית>. לדוגמה, **תגית פיסקה** היא <p> (Paragraph) **ותגית קו אופקי** היא <hr> (Horizontal Rule). אפשר לכתוב את התגיות באותיות גדולות או קטנות (אבל אנחנו נשתמש באותיות קטנות).

	הערה בכל משימה תודגש באופן בולט התוספת המתבקשת לקובץ, בהתאם למשימה הנלמדת.

לפני שאתה מתחיל

רשימת תיוג לעבודה עם ספר זה:

- מחשב PC
- מערכת הפעלה Windows 9x/Me/2000
- דפדפן Microsoft Internet Explorer
- תוכנת PWS (בהמשך תונחה היכן למצוא אותה וכיצד להתקינה).
- עורך טקסט (ראה הנחיות בהמשך).

סיומת קובץ

כדי שניתן יהיה ליצור מסמכי HTML בקלות, עליך לוודא שניתן לראות את סיומות הקבצים כאשר הינך עובד **בסייר Windows**. פעל לפי ההנחיות בהתאם למערכת ההפעלה שברשותך.

Windows 95

1. בחר בלחצן **התחל** (Start), **תוכניות** (Programs), **סייר Windows** (Windows Explorer).
2. בחר בתפריט **תצוגה** (View), **אפשרויות** (Options).
3. בחלון **אפשרויות** (Options) בחר בכרטיסיה **תצוגה** (View).
4. ודא שתיבת הסימון **הסתר סיומות קבצים**... (Hide MS-DOS file extensions...) אינה מסומנת.
5. לחץ **אישור** (OK).
6. סגור את **סייר Windows** (Windows Explorer).

Windows 98

1. בחר בלחצן **התחל** (Start), **תוכניות** (Programs), **סייר Windows** (Windows Explorer).
2. בחר בתפריט **תצוגה** (View), **אפשרויות תיקיה** (Folder Options).
3. בחלון **אפשרויות תיקיה** (Folder Options) בחר בכרטיסיה **תצוגה** (View).
4. בקבוצה **קבצים ותיקיות** (Files and Folder) ודא שתיבת הסימון **הסתר סיומות קבצים**... (Hide file extensions...) אינה מסומנת.

5. לחץ **אישור** (OK).

6. סגור את **סייר Windows** (Windows Explorer).

Windows Me

1. בחר בלחצן **התחל** (Start), **תוכניות** (Programs), **עזרים** (Accessories), **סייר Windows** (Windows Explorer).

2. בחר בתפריט **כלים** (Tools), **אפשרויות תיקיה** (Folder Options).

3. בחר בכרטיסיה **תצוגה** (View).

4. בקבוצה **קבצים ותיקיות** (Files and Folder) ודא שתיבות הסימון **הסתר סיומות קבצים**... (Hide file extensions...) אינה מסומנת.

5. לחץ **אישור** (OK).

6. סגור את **סייר Windows** (Windows Explorer).

Windows 2000 Professional

1. הפעל את **סייר Windows** (Windows Explorer).

2. בחר בתפריט **כלים** (Tools), **אפשרויות תיקיה** (Folder Options).

3. בחר בכרטיסיה **תצוגה** (View).

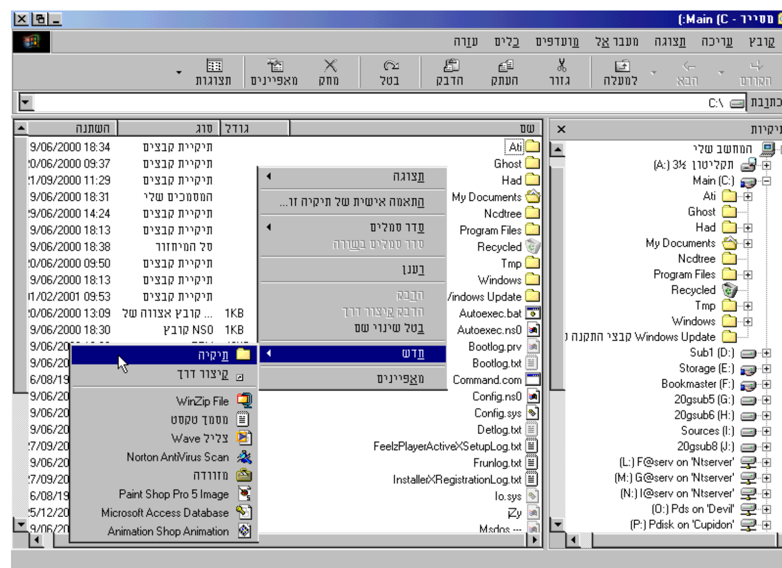
4. בקבוצה **קבצים ותיקיות** (Files and Folder) ודא שתיבות הסימון **הסתר סיומות קבצים**... (Hide file extensions...) אינה מסומנת.

5. לחץ **אישור** (OK).

6. סגור את **סייר Windows** (Windows Explorer).

מתחילים....

1. פתח את **סייר Windows** (Windows Explorer),
2. בחר בתפריט **תצוגה** (View), **פרטים** (detalis).
3. עבור לדיסק C.
4. בחלונית הקבצים (לא היכן שנמצא עץ התיקיות) לחץ לחיצה ימנית בעכבר. מתפריט הקיצור בחר **חדש** (New) ואחר כך **תיקיה** (Folder), כפי שמוצג בתרשים הבא:



תרשים 1.1

5. כאשר התיקיה החדשה בשם **תיקיה חדשה** (New Folder) מסומנת, רשום **MyHomePage** והקש **Enter** (תוכל לרשום גם שם אחר, כרצונך).
6. פתח את התיקיה שיצרת (MyHomePage או אם רשמת שם אחר), עבור לחלונית הקבצים (הריקה בשלב זה), לחץ לחיצה ימנית. מתפריט הקיצור בחר **חדש** (New) ואחר כך **מסמך טקסט** (Text Document), כמוצג בתרשים 1.2.

תרשים 1.2: יצירת מסמך טקסט חדש.

7. שים לב שהמסמך החדש, שנקרא עכשיו **מסמך טקסט חדש.txt** (New Text File.txt), מסומן.

8. שנה את שם המסמך לשם **First.html** (שים לב לשינוי הסיומת מ-txt ל-html), ולסיום הקש **Enter**. תופיע הודעה שתשאל אם ברצונך לשנות את סוג הקובץ. ענה בחיוב.

9. לאחר שינוי השם, לחץ לחיצה כפולה על שם הקובץ (לחיצה כפולה או יחידה בהתאם להגדרות Windows שאצלך). הדפדפן ייפתח והקובץ (הריק בינתיים) בתוכו, אך לא יוצג בו דבר מכיון שהמסמך (html.) ריק.

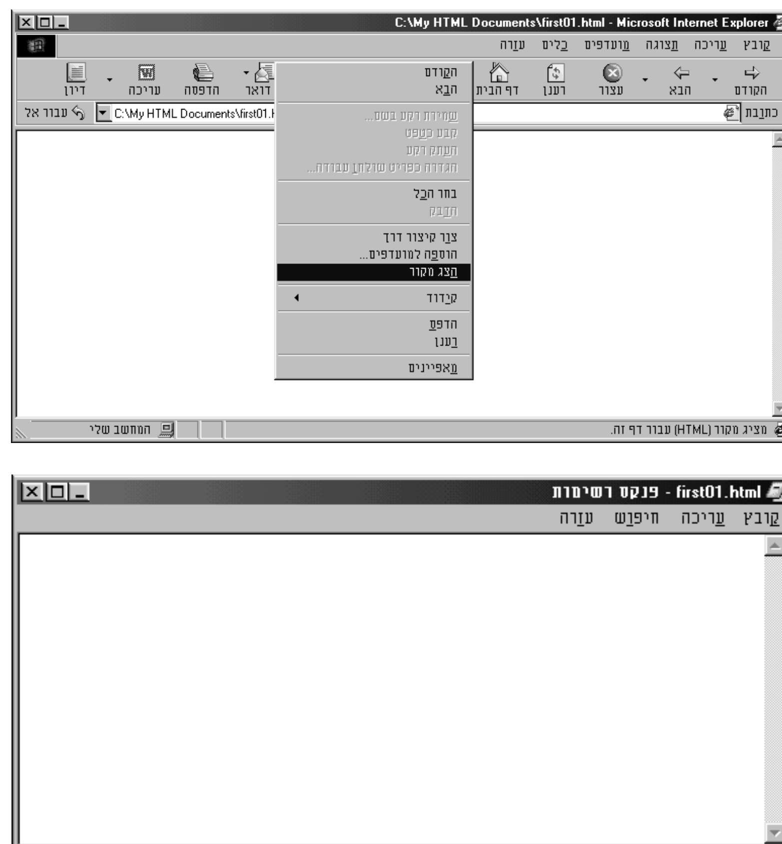
הערה

לחיצה כפולה על שם הקובץ אמורה לפתוח את הדפדפן ולהציג בעזרתו את הקובץ. אם לאחר שהצגת את סיומות הקבצים, שינית את סיומת הקובץ ל-html ועדיין לחיצה כפולה על הקובץ אינה פותחת את הדפדפן, יש לסמן את הקובץ, להחזיק לחוץ את מקש Shift, לחיצה ימנית בעכבר ובחירה בפתיחה באמצעות תפריט הקיצור. בחר מהרשימה בתוכנת הדפדפן וסמן את התיבה "**השתמש תמיד ביישום זה לפתיחת סוג קובץ זה**". לחץ על **אישור**.

אם לחיצה כפולה על קובץ עם סיומת html. פתחה את הדפדפן אבל מציגה את תוכן הקובץ, כפי שנראה בפנקס הרשימות, יהיה עליך לחזור לסעיף **סיומת קובץ** בפרק זה.

> **הערה**
>
> במהלך לימוד HTML נשתמש במסמך **first.html** (שמייד תיצור) ונשנה את תוכנו לפי שלבי הלימוד. קובץ זה בשלביו השונים ייקרא בשמות **first01.html**, **first02.html**, **first03.html** וכך הלאה. כך שבכל שלב תוכל לבדוק את עצמך.

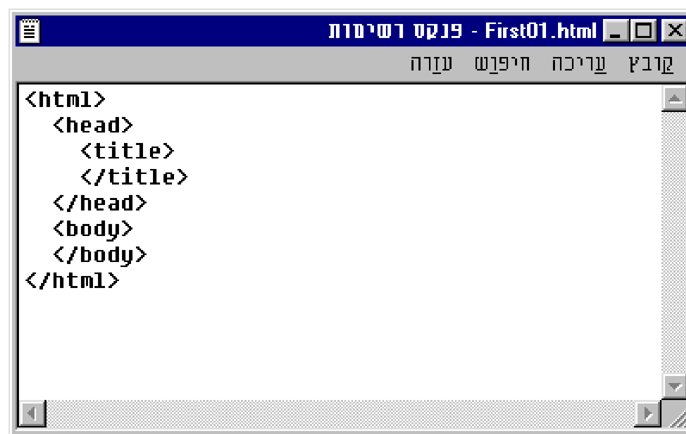
10. בדפדפן, בחלון המסמך, לחץ לחיצה ימנית ובחר **הצג מקור** (View Source) (ראה בתרשים 1.3), כדי להציג את קוד המקור של הקובץ. **פנקס הרשימות** (Notepad) ייפתח אך לא יציג דבר, מכיון שהקובץ (html.) עדיין ריק.



תרשים 1.3 : פנקס הרשימות נפתח אך עדיין אינו מציג דבר (**First.html**).

11. בחלון **פנקס הרשימות** ודא שהסמן נמצא בצד שמאל (אם לא, הקש את צירוף המקשים Ctrl+Left Shift, מקש Left Shift נמצא בצד השמאלי של המקלדת), והקלד:

```
<html>
  <head>
    <title>
    </title>
  </head>
  <body>
  </body>
</html>
```



תרשים 1.4: כך צריך להיראות חלון פנקס הרשימות כעת (**First01.html**).

לידיעתך אפשר לכתוב גם כך, ללא רווחים/ים בתחילת שורה. גם אין חשיבות כמה רווחים יש בתחילת שורה:

```
<html>
<head>
<title>
</title>
</head>
<body>
</body>
</html>
```

לדעתי, "המאמץ" בהעמדת הטקסט (הזחה) יקל בהמשך על קריאת הכתוב, וזה מה שחשוב.

12. בחר בתפריט **קובץ** (File), **שמור** (Save) כדי לשמור את הקובץ.

13. עבור לחלון הדפדפן באמצעות לחיצה על הלחצן המתאים בשורת המשימות.

14. לחץ על לחצן **רענן** (Refresh) (או הקש **F5**), עדיין לא יוצג דבר. למה? הסבר בהמשך.

15. חזור ל**פנקס הרשימות** (Notepad), באמצעות לחיצה על הלחצן המתאים שבשורת המשימות.

הוספת כותרת לחלון הדפדפן

1. חזור לחלון **פנקס הרשימות** באמצעות לחיצה על הלחצן המתאים שבשורת המשימות. בין התגית <**title**> לתגית **</title>** הקלד את הטקסט **My First HTML doc**. גם במקרה זה לא חשוב כמה רווחים יהיו מתחילת השורה, העיקר שתקליד את הטקסט בין התגית <title> לתגית </title> :

```
  <head>
    <title>
      My First HTML doc
    </title>
  </head>
  <body>
  </body>
</html>
```

2. שמור את הקובץ.
3. חזור לדפדפן ולחץ על **רענן**.
4. בשורת הכותרת של חלון הקובץ יופיע הטקסט שכתבת (ראה תרשים 1.5).



תרשים 1.5 : שם הקובץ החדש מופיע בשורת הכותרת (**First02.html**).

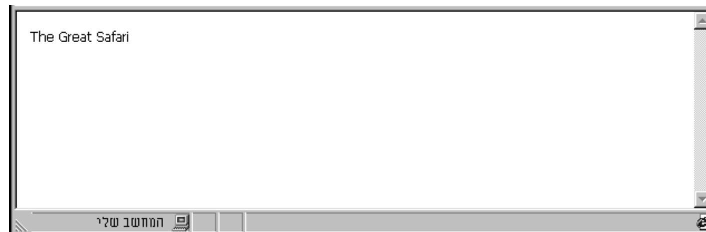
הוספת טקסט לגוף המסמך

1. חזור לפנקס הרשימות. בין התגית <**body**> לתגית **</body>** הקלד את הטקסט **The Great Safari**.

```
<html>
  <head>
    <title>
      My First HTML doc
    </title>
  </head>
  <body>
   The Great Safari
  </body>
</html>
```

2. בחר **קובץ**, **שמור** כדי לשמור את הקובץ פעם נוספת.

3. חזור לדפדפן ולחץ על **רענן**. כעת יופיע הטקסט שכתבת (כמוצג בתרשים 1.6). בשלב זה הקובץ מתחיל להיראות כדף HTML מוכר.



תרשים 1.6: הטקסט שכתבת מופיע (**First03.html**).

יצירת כותרת בדף HTML (להבדיל מיצירת כותרת לחלון הדפדפן)

1. חזור ל**פנקס הרשימות** והקף את מה שהקלדת בסעיף הקודם בתגית **<h1>**...**</h1>**, באופן הבא:

```
<html>
  <head>
    <title>
      My First HTML doc
    </title>
  </head>
  <body>
    <h1>
      The Great Safari
    </h1>
  </body>
</html>
```

הערה

תוכל גם לכתוב

```
<h1>The Great Safari</h1>
```

וזה יהיה נכון וגם ייתן את אותה התוצאה.

2. שמור את הקובץ.

3. חזור לדפדפן ולחץ על **רענן**.

4. הטקסט שתחמת בתגיות <h1>...</h1> מופיע באותיות גדולות ומודגשות (כמוצג בתרשים 1.7).



תרשים 1.7: הטקסט התחום מופיע באותיות גדולות ומודגשות (**First04.html**).

מרצף הפעולות שביצעת זה עתה, למדת שלכל תגית פתיחה יש תגית סיום. תגית פתיחה ותגית סיום נראים אותו דבר למעט התו / אותו יש להוסיף לתגית הסגירה:

- תגית **<html>...</html>** מגדירה קובץ כקובץ HTML, פותחת וסוגרת אותו.
- תגית **<head>...</head>** מגדירה מקטע בדף, שבין השאר מוגדרת בו כותרת החלון על ידי התגית <title>.
- תגית **<title>...</title>** מגדירה את כותרת החלון.
- תגית **<body>...</body>** מגדירה את מקטע גוף המסמך.
- תגית **<h1>...</h1>** מגדירה כותרת מסוג 1 במסמך.

למעשה, בנית מעין תבנית ליצירת דפי HTML.

הוספת טקסט

1. חזור לפנקס הרשימות, הקלד את הטקסט הבא: **Wild animals in the Safari** שיצטרף לגוף המסמך:

```
<html>
  <head>
    <title>
      My First HTML doc
    </title>
  </head>
  <body>
    <h1>
      The Great Safari
    </h1>
    Wild animals in the Safari
  </body>
</html>
```

2. שמור את הקובץ.

3. חזור לדפדפן ולחץ על **רענן**.

4. הטקסט שהקלדת מופיע בגוף המסמך מתחת לכותרת (כמוצג בתרשים 1.8).



תרשים 1.8: הטקסט נוסף מתחת לכותרת (**First05.html**).

הוספת רקע

1. חזור לפנקס הרשימות והוסף לתגית <body> את הפרמטרים:

```
<html>
  <head>
    <title>
      My First HTML doc
    </title>
  </head>
  <body bgcolor="Yellow">
    <h1>
      The Great Safari
    </h1>
    Wild animals in the Safari
  </body>
</html>
```

2. שמור את הקובץ.

3. חזור לדפדפן ולחץ על רענן.

4. המסמך קיבל רקע צהוב. באותו אופן תוכל לרשום במקום Yellow (צהוב): Blue (כחול), Olive (זית), Green (ירוק), Red (אדום), Teal (ירוק כהה).



תרשים 1.9: המסמך קיבל רקע בצבע שבחרת (**First06.html**).

לפני שנמשיך...

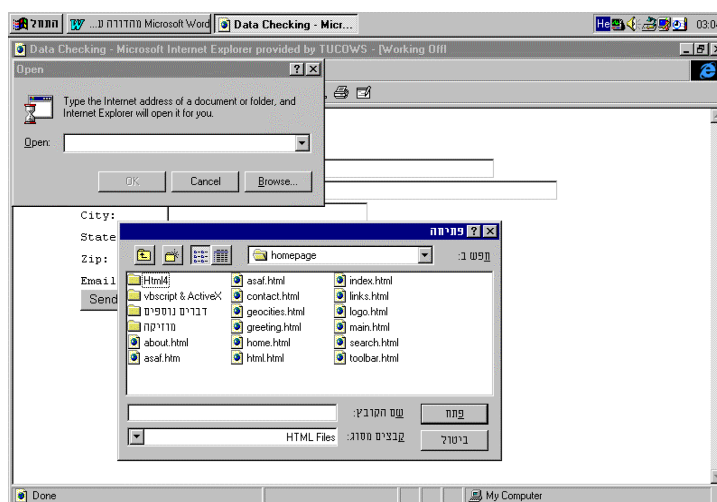
לפני שנמשיך, אני רוצה להפנות את תשומת ליבך למספר עניינים טכניים. כדי שלא תיכנס לפינות חשוכות, נעבור בקצרה על המרכיבים הבסיסיים ביצירת דפי HTML, איזה כלים נחוצים לך וכיצד תשמור את דפי ה-Web שלך.

עורך

כדי לפתח דפי HTML אתה זקוק לעורך טקסט פשוט. תוכל להשתמש ב**פנקס הרשימות** של Windows ‏(Notepad) או בעורך טקסט "עשיר", כמו ה**כתבן** (Wordpad). כשאתה שומר מסמכי טקסט שנוצרו במעבד תמלילים, כגון Word, תוכל להשתמש באפשרות **שמור כסוג** (Save As) ולבחור באפשרות **Plain Text (txt)**.

בנוסף לעורך טקסט, נחוץ לך דפדפן שבעזרתו תוכל לבדוק את תוצאות היצירה שלך. רצוי שתשתמש ב-**Microsoft Internet Explorer**.

מכיון שרוב עבודת הפיתוח תיעשה על מחשב מקומי, ולא ברשת האינטרנט, חשוב שתכיר את האפשרות לפתוח דפי HTML בעזרת תפריט File של הדפדפן.



תרשים 1.10 : קיימת אפשרות בדפדפן לפתוח קובץ HTML מהדיסק הקשיח.

בוודאי שמעת על תוכנות שיכולות לעזור לך לעצב בקליק את אתרי האינטרנט שאתה יוצר. תוכנות כאלו קיימות בשוק, ורובן אכן עושות עבודה טובה. אך למען הסדר הטוב, תיעשה הלמידה בספר זה בדרך ההקלדה הפשוטה והיסודית, כמו שכתוב בספרים.

מתן שם לקובץ

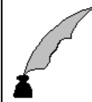
לאחר שסיימת לכתוב את דף האינטרנט בעורך הטקסט, עליך לשמור את הקובץ. רצוי ליצור תיקיה מיוחדת עבור כל אתר שאתה יוצר. בתיקיה זו תאחסן את כל הקבצים הקשורים לאתר זה, כולל תמונות. חלק ממפתחי האתרים נוהגים ליצור תיקיית משנה מיוחדת לכל סוג קובץ.

להלן החוקים למתן שמות לקבצים (טוב לכל קובץ במערכת הפעלה Windows):

- שם קובץ יכול להכיל עד 255 תווים, כולל תו רווח.
- אין להשתמש בתווים אלה: \ / * ? : < > | " ; `

דפי HTML נשמרים עם סיומת htm או html ואין הבדל בין הסיומות.

המלצה חמה: תן שמות באנגלית לקבצי HTML. מרבית השרתים באינטרנט לא "מבינים" עברית.

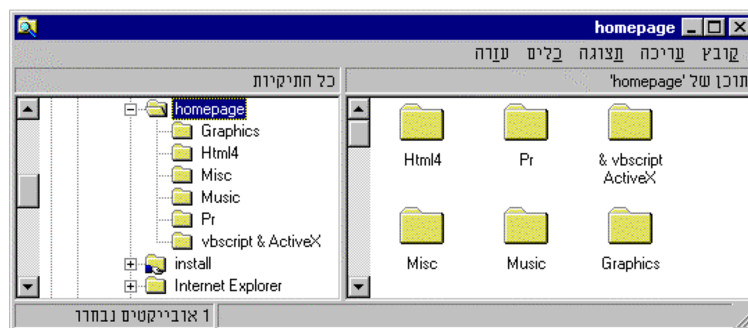


בעברית פשוטה!

שרת אינטרנט הוא מחשב המחובר ישירות לרשת, ונועד במיוחד כדי לשלוח דפי Web לדפדפני רשת המבקשים אותם. כדי לאפשר למשתמשי האינטרנט לצפות בדפים שלך, עליך לשכן אותם בשרת אינטרנט המאפשר לאנשים פרטיים או חברות לאחסן בו את האתרים שלהם.

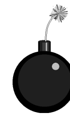
שלמות האתר

כשאתה מפתח אתר אינטרנט גדול המורכב ממספר רב של דפי HTML, תמונות וצלילים, כדאי שתשתמש במבנה היררכי של תיקיות כדי לאחסן את הקבצים. כפי שמוצג בתרשים:



תרשים 1.11: מבנה מורכב של אתר אינטרנט בו הקבצים מחולקים למספר תיקיות.

בכל מקרה, חשוב שתקפיד לעקוב בצורה מדויקת מאוד אחר השמות שאתה נותן לתיקיות שלך ואחר הקבצים הנמצאים בתוכן, כדי שלא תהיינה טעויות כלשהן ביצירת קישורים ובקריאה לקבצי תמונה וצליל. עליך לשמור על מבנה התיקיות בהן מאוחסן האתר גם כשאתה מעביר את הקבצים לשרת האינטרנט בו ישכון האתר, אחרת תסבול מתופעה מתסכלת, אך מוכרת, בה דפי האתר שלך אינם מציגים את כל התמונות, המסגרות והצלילים עליהם טרחת ועמלת זמן כה רב.



אזהרה

דפדפני אינטרנט רגישים לסיומות של קבצים. לדוגמה, אם אתה מעוניין ליצור קישור לדף HTML, הדפדפן יצפה לקובץ בעל סיומת htm או html. אם אתה מעוניין להציג תמונה בדף כלשהו, תהיה סיומת שם הקובץ בדרך כלל jpg, jpeg או gif. ניתן להציג גם קבצי טקסט בעלי סיומת txt.

פרק 2

תגיות

כותרות למיניהן לא תמיד נראות חשובות, אבל הן למעשה חלק חשוב ובלתי נפרד מעיצוב וארגון האתר שלך...

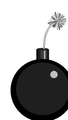
אם קראת את פרק 1 בעיון, אתה כבר יודע כיצד לשמור את הקבצים שלך בדיסק הקשיח וגם ליצור דפי HTML. בפרק זה נעמיק את הלימוד: נביט בפקודות או בתגיות (HTML Tags) בהן נשתמש לבניית אתר אינטרנט.

לאחר שתבין את המושג **תגיות**, ניגש לצעד הבסיסי ביותר בלימוד HTML: התגיות בהן תשתמש בכל דף HTML שתכתוב. תוך כדי העבודה תיווכח שאנו יוצרים תבנית בסיסית, עליה נוכל לבסס כל סוג דף באתר אינטרנט. בסיום הפרק, תוכל ליצור דף HTML מושלם מאלף ועד תו.

המונח תגיות

בעולם הפיתוח באינטרנט תיתקל פעמים רבות במונח **תגיות HTML**. גם בספר זה תיתקל במונח זה פעמים רבות. **תגיות** הן הפקודות הבסיסיות ביותר עליהן בנויה הכתיבה ב-HTML. כמעט כל דבר בדפי Web מושתת על התגיות.

תגיות תופענה בקוד HTML תמיד בתוך סוגריים זוויתיים: <תגית>. לדוגמה, **תגית פיסקה** היא <p> (Paragraph) **ותגית קו אופקי** היא <hr /> (Horizontal Rule). אפשר לכתוב את התגיות באותיות גדולות או קטנות, כך <HR />, <hR />, <Hr />, <hr /> הן אותה תגית (בספר זה, נשתמש באותיות קטנות בעת כתיבת תגיות, וזאת מתוך ראייה קדימה של התפתחות התקנים בעולם האינטרנט: XHTML, XML ואחרים, הדורשים תגיות באות קטנה).



אזהרה

הדפדפן לא יתריע על חוסר בתגית או על תחביר לא נכון. לכן, חשוב להקפיד ולסגור את הסוגריים בכל תגית ותגית. במקרים רבים, במיוחד בתחילת הדרך, תיתקל בבעיות שונות ומשונות ורק בגלל סימן גדול מ- (>) או קטן מ- (<) ששכחת לסגור, או בגלל תגית שרשמת לא נכון או לא במקום. כדאי להקפיד על כך.

קדימה לעבודה

לפני תחילת העבודה הרצינית, עליך להתכונן. צור תיקיה המיועדת לאחסון קבצי האתר שאתה יוצר, למשל **C:\My WebSite**. תן לתיקיה שם קצר וקולע. כך תוכל להבטיח לעצמך פחות הקלדות ותמיד תדע היכן הקבצים שלך.

1. פתח את **סייר Windows** (Windows Explorer).
2. צור תיקיה חדשה.
3. פתח את התיקיה שיצרת.
4. עבור לחלונית הקבצים (הריקה בשלב זה).
5. לחץ לחיצה ימנית ומתפריט הקיצור בחר **חדש** (New), **מסמך טקסט** (Text Document).
6. סמן את הקובץ שיצרת, שנקרא עכשיו **מסמך - טקסט חדש.txt** (New Text File.txt).
7. שנה את שם המסמך ל-**MyFirstHTML.html** (שים לב לשינוי הסיומת מ-txt ל-html). תופיע הודעה שתשאל אם ברצונך לשנות את סוג הקובץ, ענה בחיוב.
8. לאחר שינוי השם, לחץ לחיצה כפולה על שם הקובץ (כפולה או יחידה בהתאם להגדרות Windows שאצלך). הדפדפן ייפתח והקובץ (הריק בינתיים) בתוכו, אך לא יוצג בו דבר מכיון שהמסמך (html.) ריק.
9. בדפדפן, בחלון המסמך, לחץ לחיצה ימנית ובחר **הצג מקור** כדי להציג את קוד המקור של הקובץ. פנקס הרשימות ייפתח אך לא יציג דבר מכיון שהקובץ (html.) עדיין ריק.
10. בחלון **פנקס הרשימות** ודא שהסמן נמצא בצד שמאל (אם לא, הקש את צירוף המקשים **Ctrl+Left Shift** שבצידה השמאלי של המקלדת), והקלד:

```
<html>
  <head>
    <title>
    </title>
  </head>
  <body>
    Hello!
    <hr />
  </body>
</html>
```

11. בחר בתפריט **קובץ**, **שמור** כדי לשמור את הקובץ.
12. עבור לחלון הדפדפן ולחץ על לחצן **רענן** (או הקש **F5**).

טיפ

אם אתה מעוניין לכתוב פיסקה מיושרת לשמאל (משמאל לימין) בפנקס רשימות, עליך להקיש את צירוף המקשים Ctrl+Left Shift שבצידה השמאלי של המקלדת. כדי לכתוב פיסקה מיושרת לימין (מימין לשמאל), הקש את צירוף המקשים Ctrl+Right Shift שבצידה הימני של המקלדת.

אז למה בשביל המילה Hello, סימן קריאה וקו אופקי אנו זקוקים להקליד כל כך הרבה טקסט? לא חייבים. HTML מאוד סלחנית ואינה מחייבת באופן, שכדי לקבל את המילה Hello, סימן קריאה וקו אופקי היית יכול להקליד רק את המשפט הבא:

```
Hello!<hr />
```

ואפילו אין צורך בהקשה על Enter, כי גם אם תקיש לא תהיה לכך כל משמעות או השפעה על מראה הטקסט בדפדפן. HTML לא מתייחסת לרווחים ולא להקשות Enter.

והערה נוספת באותה עניין של אי הפורמליות של כתיבת HTML. הקובץ להצגת המילה Hello, סימן קריאה וקו אופקי היה יכול להיראות כמו בצד שמאל, אך אני מעדיף כתיבת הקוד כמו בצד ימין, שהיא נוחה יותר לקריאה ולהבנה:

```
<html>
  <head>
    <title>
    </title>
  </head>
  <body>
    Hello!
    <hr />
  </body>
</html>
```

```
<html><head><title></title></head>
<body>Hello!<hr /></body></html>
```

פתיחת דף הנמצא בדיסק

כעת, רק לשם ההנאה, פתח את הדפדפן. אם אתה מתבקש להתחבר לרשת, לחץ על **ביטול** - Cancel (אם תראה כי קיימת בעיה בטעינת הקובץ מכיון שאינך מחובר, סימן שהדפדפן שלך לא מאפשר הצגת קבצים ללא חיבור לרשת). כעת, פתח את הקובץ **MyFirstHTML.html** שיצרת קודם לכן בעזרת תפריט **קובץ** (File) (שבדפדפן).

תוכל גם לפתוח אותו באמצעות **סייר Windows**: אתר את התיקיה, פתח אותה, אתר את הקובץ ולחץ עליו לחיצה כפולה.

לאחר שבחרת את הקובץ **MyFirstHTML.html**, עליך לאשר את פעולת הפתיחה. אם הכל כשורה תראה כעת את הדף הראשון שיצרת. זה אמנם רק !Hello עם קו אופקי מתחתיו, אבל זו התחלה. חשוב לציין, כי מכאן והלאה הדפים יהיו מורכבים יותר. חשוב מכך, הם גם ייעשו מושכים יותר. בתרשים 2.1, תוכל לראות את התוצאה שאמורה להופיע אצלך בדפדפן.



תרשים 2.1: תוכל להשתמש בדרך עבודה זו כדי לבדוק את כל הדפים שלך.

חשוב שתזכור את הפרטים הבאים:

- המילה "Hello" לא צריכה שום תוספת של קוד. דפדפני האינטרנט יודעים להציג טקסט פשוט לפי הגופן שמוגדר בהם כברירת מחדל. בדרך כלל הגופן Times New Roman בגודל 12, אלא אם המשתמש הגדיר אחרת.
- התגית <hr />, שפירושה **קו אופקי** (Horizontal Rule), מציירת קו אופקי. התגית מעובדת על ידי הדפדפן ומוצגת בהתאם.
- אתה חייב לשמור את קבציך עם סיומת **html** או **htm**, אם אתה רוצה שהם יוצגו כראוי. אם תשמור את הקובץ בעורך הטקסט שלך כקובץ עם סיומת txt או asc, הדפדפן יציג את תוכן הקובץ ולא יציג קו אופקי (<hr />).

כותרת לדף HTML

בפרק 3, נציג בפניך מידע מעמיק על HTML. בינתיים, רק נראה איזה הבדל עושים שינויים קטנטנים לקובץ שלנו **MyFirstHTML.html**.

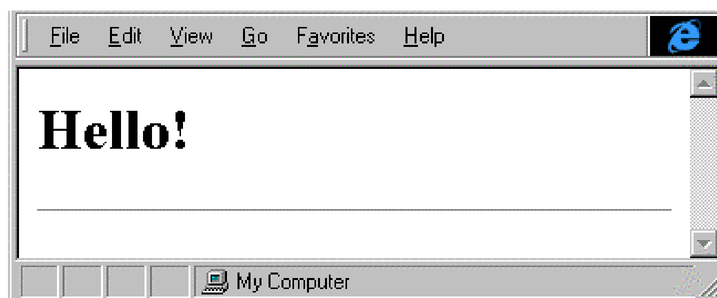
פתח את הקובץ **MyFirstHTML.html** בעורך הטקסט שלך (קרוב לוודאי שהוא כבר פתוח, אם אתה מתרגל את הפרק הזה ברצף). כעת, הקלד <h1> לפני המילה Hello ו-</h1> אחריה. אין שום צורך להזיז את <hr /> ממקומו.

הקוד צריך להיראות כך:

```
<html>
 <head>
  <title>
  </title>
 </head>
 <body>
  <h1>Hello!</h1>
  <hr />
 </body>
</html>
```

כעת, שמור את הקובץ בשם **MyFirstHTMLPlus.html** (עשה זאת בעזרת הפקודה **קובץ** (File), **שמור בשם** (Save As)) תוכל לטעון את הקובץ החדש דרך הדפדפן, תפריט **קובץ** (File) או פשוט ללחוץ עליו לחיצה כפולה מתוך **סייר Windows**. כעת מופיעה המילה "Hello!" בצורה מודגשת וגדולה בהרבה מאשר בגירסה הקודמת של המסמך.

אז מה היה לנו שם? בעזרת תוספת קטנה של <h1> לפני ו-</h1> אחרי, גרמנו למילה "Hello!" להיות כותרת ברמה 1. הדפדפן שלך תרגם את הפקודות <h1>...</h1> ככותרת ושייך את התכונה הזו לטקסט שבין התגיות, המילה "Hello!". בדרך כלל, כותרת h1 גדולה בהרבה מגודל תצוגה של טקסט רגיל, אך שוב, התצוגה תלויה בהגדרות המשתמש בדפדפן שלו. בתרשים 2.2, תוכל לראות את התוצאה:



תרשים 2.2: תגיות כמו <h1></h1> משפיעות על הטקסט הנמצא ביניהן.

מה שעשינו כעת למסמך הוא שינוי קטן, המייצג את אחד העקרונות החשובים ביותר ב-HTML.

תיחום תגיות

כל תגית html צריך שתהיה לה תגית פתיחה ותגית סיום. לדוגמה לתגית הכותרת <h1>, יש תגית סגירה </h1>. יש תגיות שלא הוגדרה להן תגית סגירה, למשל: <hr>,
, <frame>, <img> ואחרות. ההתאמה לתקן XHTML כוללת הוספת התו בסיום התגית. את התגית <hr> נכתוב מעתה <hr /> ואת התגית
 נכתוב
 וכך הלאה. למרות שזה לא חובה "לסגור" תגיות שאין להן תגית סגירה, אנחנו נסגור.

כדי שתוכל להבחין בין סוגי התגיות, אנו נציין תגית שיש לה תגית פתיחה ותגית סגירה כך: <head> ותגית שיש לה רק תגית פתיחה נכתוב כך: <meta />.

חשוב לזכור את הנקודות הבאות:

- לאחר הופעת תגית פתיחה, תופיע גם תגית סיום מתאימה. תגית הסיום זהה לתגית הפתיחה, אך כוללת תוספת תו (/). לדוגמה, תגית הפתיחה של טקסט מודגש היא ותגית הסיום היא .
- תגיות שאין להן תגית סגירה יש לכתוב כך <hr /> ולא <hr>.

- אין צורך ברווחים בין התגיות וגם לא בהקשות Enter. HTML לא מתייחסת להקשות אלו. בכל מקרה של מספר רווחים ברצף, הדפדפן מתייחס אליהם כתו רווח אחד. הדפדפן יציג את הטקסט שלך בדיוק באותה הצורה. בשלב זה, עדיין אינך מכיר את הדרכים לשליטה על רווחים. לכן, תן לדפדפן לעשות את העבודה בשבילך.

✓	**טיפ** למעשה, הדפדפן מתעלם מכל רווח ומכל מעבר שורה. מבחינתך, זה עשוי להיות יתרון גדול מכיון שאתה יכול להשתמש ב-Enter כדי להפריד בין התגיות. בדרך זו יהיו לך מסמכים מסודרים יותר ונוחים יותר לקריאה ולעדכון, בעוד שבדפדפן הם ייראו בדיוק אותו הדבר.

- לאחר שהתחלת לערוך מסמך HTML, אתה יכול לחזור אליו ולערוך בו שינויים. כל מה שעליך לעשות כדי לראות את הגירסה המעודכנת בדפדפן, הוא ללחוץ על לחצן **רענן** (Refresh).

✓	**טיפ** הוסף את תגית ה**סגירה** (</title>) מייד לאחר תגית ה**פתיחה** (<title>) ורק אחר כך חזור לרשום את תוכן התגית. כך, תחסוך מעצמך כאבי ראש מיותרים ותוכל להיות בטוח שלא שכחת להציב תגית סיום.

לאחר שהבנו את המונח **תגיות HTML**, אנו יכולים להתקדם וללמוד על תגיות המסמך הבסיסיות. תוך כדי כך, ניצור לעצמנו תבנית עליה נבנה את כל המסמכים בעתיד.

לכל דף HTML יש אותו מבנה בסיסי

לפנינו אחת מאבני היסוד של כל מסמכי HTML, ראה קובץ **Template.html**. כל מסמך שאנו יוצרים, יהיה בנוי באותו מבנה בסיסי:

```
<html>
  <head>
    <title>
      The Title of Your Page
    </title>
  </head>
  <body>
    The web page itself, containing all the links,
    graphics, text &amp; so on.
  </body>
</html>
```

מכיון שזהו הבסיס לכל דפי HTML שתיצור, תוכל ליצור לעצמך קובץ תבנית ובעזרתו ליצור את כל הקבצים החדשים שלך, ללא צורך בכתיבת הפקודות הבסיסיות המופיעות בקובץ זה. כדי לבנות לעצמך תבנית, פתח את עורך הטקסט שלך והקלד לתוכו את הפקודות המופיעות למעלה. לאחר מכן, שמור את הקובץ בשם **Template.html**. בכל פעם שתרצה ליצור דף HTML חדש, כל שיהיה עליך לעשות הוא לפתוח את הקובץ **Template.html** בעורך הטקסט ולשמור אותו בשם אחר. על הקובץ החדש שיצרת תוכל לעבוד כקובץ לכל דבר.

התגית <html>

התגיות המגדירות את המסמך כמסמך HTML (<html>...</html>) תוחמות ביניהן את כל התגיות והטקסט שבמסמך עצמו. תגית ה**פתיחה** (<html>) היא התגית הראשונה במסמך, ותגית ה**סיום** (</html>) היא התגית המסיימת את המסמך.

התגית <head>

תגית **<head>** ותגית הסיום **</head>** מגדירות בתוכן את כותרת המסמך. טקסט הנמצא בתחום זה מכיל מידע שאינו מוצג בדפדפן כחלק מהמסמך.

בתחום התגיות <head>...</head> יכולות להופיע מספר תגיות נוספות:

<title>...</title>	כותרת המסמך.
<meta />	מכיל מידע נוסף על המסמך.

חשוב להכיר תגיות אלו, מכיון שהן עוזרות לבנות מסמכי HTML עשירים יותר במידע ומאפשרות לבנות אתרי אינטרנט מתוחכמים יותר. כדי להיות מוכן לדרישות הטכנולוגיות, חשוב להכיר מקרוב את התגיות הכלולות בכותרת.

התגית <title>

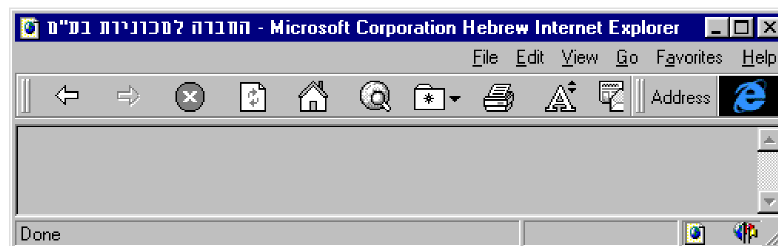
לכל דף שתיצור חייב להיות שם **אחד**, לא פחות ולא יותר. שם הדף יופיע בין תגית <title> לתגית </title>. רצוי לתת לכל דף שם המתאר אותו בצורה הטובה ביותר.

טיפ

השם של כל מסמך HTML מהווה את כותרת החלון. חשוב מאוד להשקיע מחשבה בשם הדף, מכיון שבתהליך הדפדוף בין האתרים בהם כבר ביקר המשתמש ובגישה ל**מועדפים** (Favorites), כל מה שהוא רואה זה את השם שבמסגרת התגית <title>.

תרשים 2.3 מראה את הטקסט בכותרת החלון. הטקסט נכתב ב-HTML כך, כפי שמופיע בקובץ **MyCar.html**:

```
<title>החברה למכוניות בע"מ</title>
```



תרשים 2.3: שם האתר מופיע בכותרת החלון.

התגית <meta />

בעזרת תגית זו ניתן לציין מידע לגבי המסמך אותו יצרת ובין השאר לציין את השפה בה כתוב המסמך. תגית זו תתוארת בהמשך בפרק העוסק בכתיבת עברית.

פרק 3

טקסט ועיצוב

החלק העיקרי של דפי האינטרנט הוא הגוף (<body>). לא חשוב אם גוף המסמך גדול או קטן, העיקר שיהיה נכון!

התגית <body>

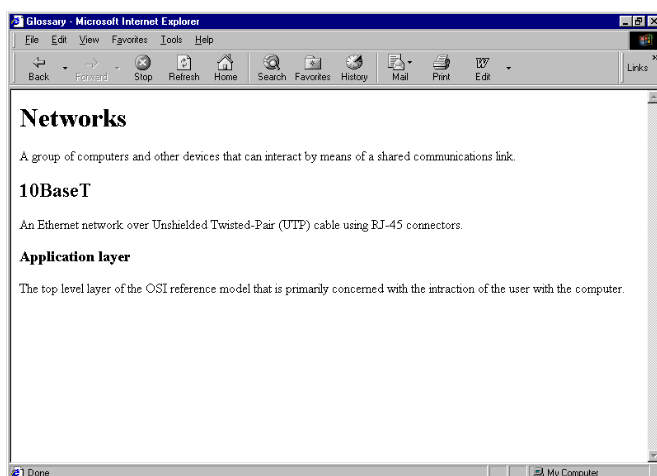
גוף כל מסמך ברשת מוגדר על ידי התגית **<body>**. קיימת תגית פתיחה <body> ממנה מתחיל גוף המסמך, ותגית </body> המסמנת את סיום המסמך. בתוך גוף המסמך ייכללו רוב מרכיבי המסמך, כגון טקסט, גרפיקה, קישורים, כותרות, טפסים, מפות תמונה, טבלאות וכל מה שיכול המבקר באתר לראות בדפדפן. להלן דוגמה כיצד נראית התגית <body> במסמך HTML. ראה קובץ **Template.html**:

```
<html>
  <head>
    <title>
      The Title of Your Page
    </title>
  </head>
  <body>
    The web page itself, containing all the links, graphics, text &amp; so on.
  </body>
</html>
```

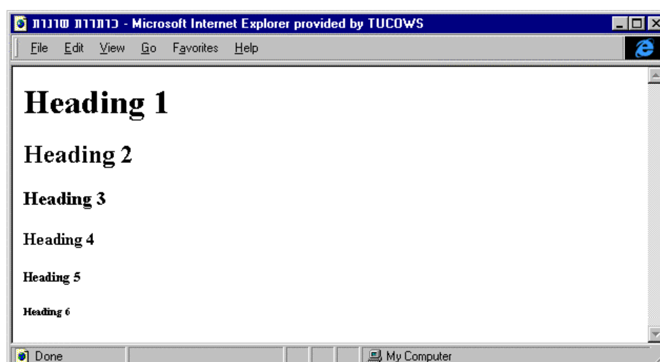
כפי שתראה מייד, התגית <body> היא בעלת מיקום ברור וקבוע במסמך. במסמך הבא מוטמעת תגית זו בין התגיות <html>. מכאן אנו למדים כי התגית <body> היא תת-מבנה של <html>. כמעט כל דבר בעל חשיבות במסמך יהיה חלק מהתגית <body>.

כותרות, התגית <h1>

הדבר הראשון, פחות או יותר, שמפתחי HTML לומדים לכתוב במסמך הוא כותרות. קיימות שש רמות של כותרות, עבור כל רמת כותרת - דפדפני האינטרנט מציגים גודל אות שונה. ניתן להשתמש במספר סוגי כותרות כדי לדרג את מבנה הנתונים במסמך. הרעיון הוא לחלק את המסמך למספר רמות של כותרות, כשלכל רמה מייצגת רמת חשיבות של הנתונים המוצגים בה. תרשים 3.1 מציג דוגמה לדירוג שכזה, קובץ **Header1.html**:



תרשים 3.1: כותרות הן כלי מצוין לחלוקת המסמך לרמות.



תרשים 3.2: כך מציג Internet Explorer את רמות הכותרות השונות (קובץ **Header2.html**).

תקן HTML תומך בשש רמות של כותרות: h1, h2, h3, h4, h5 ו-h6. כל רמת כותרת מוצגת על ידי הדפדפן בדרך שונה, כך שכותרת h4 לא תיראה כמו כותרת h1. ראה קובץ **Header2.html**. חשוב לזכור שקיימים הבדלים בין דפדפנים, וכי לעיתים ההבדלים בין הכותרות תלויים גם בהעדפות המשתמש לגבי גופנים (סוג וגודל). לכן, רצוי שלא תשקיע מחשבה רבה מדי לגבי עיצוב הכותרות.

לתגית <h1> ניתן להוסיף את המאפיין align :

מאפיין	משמעות	ערכים
align	יישור	left, center, right, justify

המשפט הבא, ימרכז את הכותרת בתגית <h2> :

```
<h2 align="center">Thank you for stopping by.</h2>
```

פיסקה, התגית <p>

התגית **<p>...</p>** תוחמת קטע של טקסט לפיסקה, כדי לאפשר קריאה נוחה ומראה משופר. למרות שהתקן הנוכחי של HTML מגדיר את תגית <p> כ**תגית תחום** הדורשת תגית פתיחה ותגית סיום, הדפדפנים עדיין מזהים את התקן הקודם של תגית זו, שדרש רק תגית אחת (תגית ריקה). למרות זאת, רצוי להשתמש בצורה העדכנית ביותר של תגית <p>, רק כדי למנוע בעיות שעלולות להופיע בעתיד. הדרך המומלצת ביותר לשימוש בתגית היא :

```
<p>Thank you for stopping by.</p>
```

ראה לדוגמה שימוש בתגית הפיסקה במסמכי HTML (במקרה זה **Header1.html**) :

```
<h1>Networks</h1>
  <p>
   A group of computers and other devices that can
   interact by means of a shared communications link.
  </p>
```

לתגית <p> ניתן להוסיף את המאפיין align :

מאפיין	משמעות	ערכים
align	יישור	left, center, right, justify

המשפט הבא, ימרכז את הפיסקה :

```
<p align="center">Thank you for stopping by.</p>
```

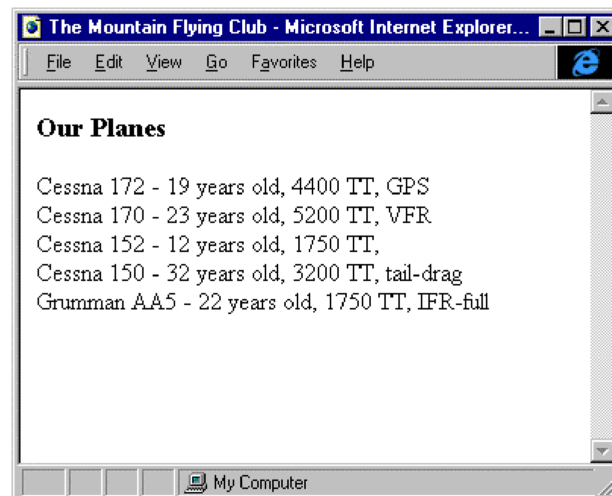
מעבר שורה, התגית

לעיתים, אתה מעוניין שהטקסט יעצור בנקודה כלשהי וימשיך בשורה הבאה, ללא רווחים. לשם כך תשתמש בתגית **
**. תגית זו היא תגית ריקה ומופיעה במקום בו תרצה מעבר לשורה חדשה. תגית זו מורה לדפדפן להציג את הטקסט בשורה חדשה.

ההשפעה של תגית
 דומה למעבר שורה בודד במעבד תמלילים. בניגוד למעבר פיסקה, בעזרת התגית <p>...</p>, הנותן רווח הדומה יותר לרווח כפול במעבד תמלילים או להקשה על מקש Enter בסוף פיסקה. אין הגבלה על מספר תגיות

בהן ניתן לעשות שימוש. זו הדרך המדויקת ביותר להגדרת גלישת השורות בעת הצגת הדף, וליצירת טקסט מעוצב בקפידה.

תרשים 3.3, המציג את קובץ **Br.html**, מדגים שימוש בתגית
 כדי ליצור רשימות טקסט.

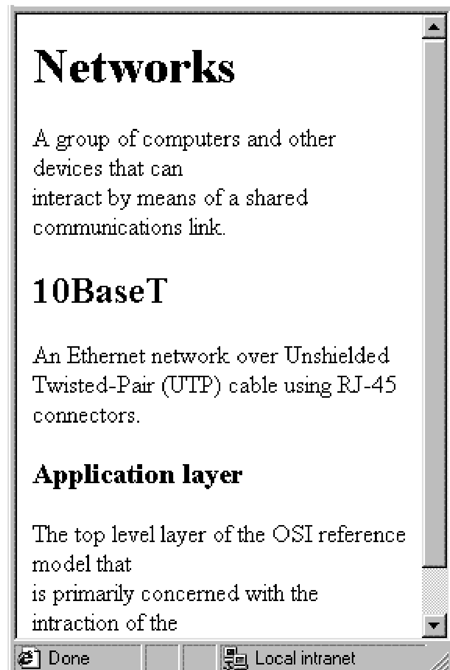


תרשים 3.3: בכל השורות נעשה שימוש בתגית
 כדי לאלץ מעבר שורה.

בקובץ **Br.html** תוכל להתרשם מהקוד שעליו בנוי המסמך לעיל:

```
<html>
  <head>
    <title>
    </title>
  </head>
  <body>
    <h3>
      Our Planes
    </h3>
    Cessna 172 - 19 years old, 4400 TT, GPS<br />
    Cessna 170 - 23 years old, 5200 TT, VFR<br />
    Cessna 152 - 12 years old, 1750 TT, <br />
    Cessna 150 - 32 years old, 3200 TT, tail-drag<br />
    Grumman AA5 - 22 years old, 1750 TT, IFR-full<br />
  </body>
</html>
```

בתרשים 3.4, לעומת זאת, תוכל לראות מה קורה כשמשהו משתבש (הפעל את קובץ **Header3.html**). לעיתים, גם אם אתה מאוד משתדל ליצור מסמך נאה למראה, עדיף לתת לדפדפן לעשות את מעברי השורות. אם תשתמש בתגית
 במשפטים ארוכים מדי, אתה צפוי לצרות צרורות.



תרשים 3.4 : Internet Explorer מבצעת את פעולת התגית
 באופן אוטומטי במקרה והטקסט הגיע לקצה החלון (ראה פיסקה אמצעית) (**Header3.html**). בפיסקה העליונה והתחתונה תוכל לראות מה קורה לטקסט שעוצב על ידי
 במידה וגודל החלון שונה מהגודל שהמעצב תכנן.

טקסט מעוצב מראש

אם יש לך טקסט כתוב מראש, אותו אתה מעוניין להכניס לתוך מסמכי HTML שלך, תוכל לחסוך לעצמך הקלדה מיותרת. בעזרת תגית התחום <pre>...</pre> תוכל להוסיף את הטקסט המעוצב ולשמור על עיצובו גם בתוך מסמך HTML. מעברי שורות שנעשו באמצעות הקשה על Enter בתגית זו נשמרים, וכך נמנע שימוש בתגית
. תגיות תחום נוספות, כגון <blockquote>...</blockquote>, מאפשרות לך שליטה רבה מאוד על המראה המדויק של המסמך.

תחום, התגית <pre>

מרכיב הטקסט המעוצב מראש המיוצג על ידי התגיות **<pre>...</pre>**, תומך בחללים ריקים למיניהם ומאפשר לתגיות עיצוב הטקסט להשפיע על הטקסט המעוצב מראש. החיסרון היחיד הוא, שדפדפנים נוטים להציג קטעים אלה בגופן משעמם (בדרך כלל, Courier). בדפדפנים מסוימים ניתן לשנות זאת.

חשוב שאתה צריך לבנות מסמך אינטרנט שיציג שירי משוררים. כך תציג את הטקסט מסודר בצורה יצירתית ללא קושי (קובץ **Pre.html**):

```
<pre>
   A little bit of text in here
      A little bit of text over here
   And some more text in here
      And a little bit more text in here
</pre>
```

כשאתה מעוניין להציג קטעי קוד של תוכנית מסוימת בדף ה-Web שלך, תוכל לעשות זאת באמצעות טקסט מעוצב מראש. בנוסף, תגיות <pre>...</pre> מאפשרות לך ליצור טבלאות טקסט מדויקות, למרות שלדעת רוב המפתחים ב-HTML עדיף השימוש ביכולת יצירת הטבלאות של HTML עצמה.

תגית <blockquote>

מעוניין להגדיל כניסה של פיסקה מסוימת? ובכן, לתגית **<blockquote>** התשובה. תגית זו אינה שומרת את מבנה הרווחים והפרדת השורות, אלא יוצרת הגדלה אחידה של כניסת הטקסט בפיסקה.

ניתן לשלב תגיות HTML נוספות בתוך <blockquote>, כמו תגיות לעיצוב טקסט והפרדת שורות. בדוגמה הבאה תוכל לראות כיצד להשתמש בתגית זו (קובץ **BlockQuote.html**):

```
<blockquote>
    "Jonathan's life has changed dramatically in the past
    couple of years, he is not the same person anymore.
    The price of succes has been too high for him. I
    think he had better off staying anonymouse." - Rich
    Peterson a friend.
</blockquote>
```

עיצוב טקסט

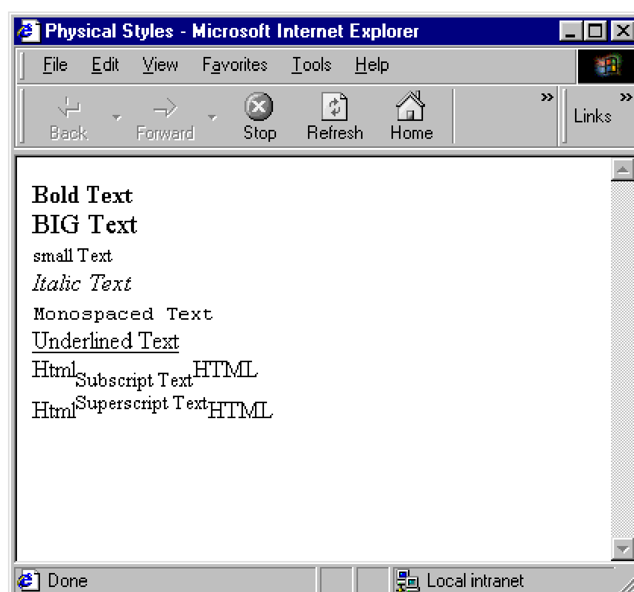
אם אי-פעם השתמשת במעבד תמלילים ראוי לשמו, תזהה מייד את עיצובי הגופנים של HTML. עיצובים אלה מדגישים את הטקסט בעזרת **הדגשה**, *הטיה*, קו תחתון ועוד. תגיות תחום אלו הן כלליות ויוצגו בדרך דומה בכל הדפדפנים הקיימים כיום בשוק.

בטבלה הבאה מוצגים עיצובי הטקסט הבסיסיים האפשריים בעת יצירת דפי Web. בתרשימים הבאים ובקובץ **Format.html** תוכל להתרשם מהשימושים הנעשים בהם.

עיצוב גופנים ומשמעותם

עיצוב	משמעות
<b>	הדגשה
<big>	אותיות גדולות
<small>	אותיות קטנות
<i>	הטיה
<tt>	מכונת כתיבה
<u>	קו תחתון
<sub>	כתב תחתי
<sup>	כתב עילי

בקובץ **Format.html** רוכזו מספר עיצובי טקסט.



תרשים 3.5: טקסט מעוצב.

גופן, התגית <font>

ניתן לקבוע את גודל הגופן בתוך הדפדפן. את הגודל ניתן לדרג לפי גדלים מ- 1 עד 7, כאשר האפשרויות הן בין (+4) ל-(-4). להלן המאפיינים של התגית **<font>**:

מאפיין	משמעות	ערכים
size	גודל הגופן	1, 2, 3, 4, 5, 6, 7 or +1, +2..
color	צבע הגופן	blue, green ... #rrggbb
face	שם הגופן	Arial, Times new roman,...

הערה

בכל הקשור לצבע וגם לסימון #rrggbb פנה לפרק 6.

המאפיין הינו תכונה. במקרה של תגית <font> התכונה יכולה להיות גודל, צבע ו/או סוג. שים לב שהשימוש במאפיין כולל שני מרכיבים:

1. שם המאפיין, כמו size, color וכדומה.
2. ערך המאפיין הנרשם בין גרשיים, למשל הערך "1" למאפיין size.

קובץ **FontDemo.html** מדגים את השימוש בתגית זו במספר מצבים:

```
<html>
 <head>
  <title>
  </title>
 </head>
 <body>
  This is the Default font<br />
  <font size="-1">Size = -1</font> <br />
  <font size="-2">Size = -2</font> <br />
  <font size="-3">Size = -3</font> <br />
  This is the Default font <br />
  <font size="+1">Size = +1</font> <br />
  <font size="+2">Size = +2</font> <br />
  <font size="+3">Size = +3</font> <br />
  <br />
  <font size="6">W</font><font size="-1">elcome</font>
  <br />
  <font size="3" color="blue">Blue is the sky</font><br />
  <font color="red">Red is the blood</font><br />
  <font size="+1" color="green">Green is the grass</font>
```

```
  </body>
</html>
```

הדוגמה הבאה יוצרת אות ראשונה מוגדלת:

```
<font size="6">W</font><font size="-1">elcome</font>
```

שים לב לתגית הסגירה . לכל תגית חייבת להיות תגית סגירה . אם אין תגית סגירה, העיצוב שהוגדר תחת התגית <font> יחל מכאן ועד סוף הדף.

המאפיין size המוגדר תחת התגית <font> יכול להיות מספר בתוספת סימן + או -. המשמעות היא הגדלה או הקטנה של גודל הגופן יחסית לגודל האחרון שבשימוש.

```
<font size="+2">Size = +2</font>
```

המאפיין color נותן צבע לטקסט. שורה זאת מגדירה את הטקסט שבין התגית לתגית בצבע כחול.

```
<font size="3" color="blue">Blue is the sky</font>
```

בהמשך תלמד על צבעים. כרגע תוכל להשתמש בשמות כמו red, blue, green, yellow, navy, aqua וכמובן white, black ו-fuchsia אותם תוכל לכתוב באותיות גדולות או קטנות כרצונך.

המאפיין face קובע את סוג הגופן שיהיה בשימוש. ללא הגדרה זו ייעשה שימוש בגופן ברירת המחדל שמוגדר בדפדפן (כל משתמש יכול לשנות הגדרות אלה). במידה ומוגדרים מספר גופנים, ייעשה שימוש בגופן הראשון שיימצא במחשב מתוך הרשימה. בדוגמה שלפניך, ייעשה שימוש בגופן verdana אם הגופן arial לא יימצא, אז ייעשה שימוש בגופן san-serif ואם גם הוא לא יימצא אז הדפדפן ייעשה שימוש בגופן ברירת המחדל.

```
<font size="3" color="blue" face="arial, verdana, san-serif">Blue is the sky</font>
```

גופן בסיסי, התגית <basefont />

כדי לקבוע את הגופן הבסיסי למסמך כולו, השתמש בתגית **<basefont />**. גם לתגית זו יש אותם המאפיינים כמו לתגית <font>, והם:

מאפיין	משמעות	ערכים
size	גודל הגופן	1, 2, 3, 4, 5, 6, 7 or +1, +2..
color	צבע הגופן	blue, green ... #rrggbb
face	שם הגופן	Arial, Times new roman,...

ברירת המחדל של הדפדפן היא בדרך כלל גופן בגודל 3 ("size="3) בצבע שחור ("color="black). בעזרת התגית <basefont /> ניתן לקבוע גודל גופן אחר כמו גם צבע אחר וגם סוג אחר, למשל.

```
<basefont size="2" color="navy" face="desdemona" />
```

את פעולת התגית <basefont /> תוכל לראות בקובץ **BaseFont.html**.

תוכל לשנות בעצמך את הערכים בתגית <basefont />. למשל, במקום "size="2 רשום "size="6 וראה מה קורה. במקום "color="navy רשום "color="yellow ובחן את התוצאות, ובמקום "face="desdemona רשום "face="arial.

אם מוגדרת התגית <basefont />, אז המאפיין size בשילוב הסימן + או - בתגית <font> מתייחס לגודל שנקבע בתגית <basefont />.

קו אופקי, התגית <hr />

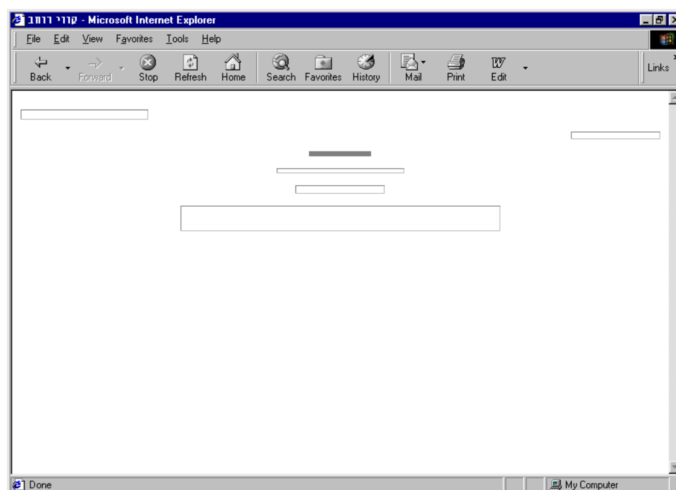
התגית **<hr />** יוצרת קו אופקי עם הצללה, לכל רוחב מסך הדפדפן. כשהמשתמש משנה את גודל המסך, משתנה גודל הקו האופקי בהתאם. התגית <hr /> יוצרת מעברי פיסקה לפניה ולאחריה. ראה קובץ **Header4.html**:

להלן מספר הגדרות מעניינות וחשובות לתגית <hr />. תוכל לקבוע את עובי הקו, אורכו ומיקומו בתוך חלון הדפדפן. כמו כן, אתה יכול לוותר על המראה המוצל של הקו לטובת מראה של קו שחור יציב. הטבלה הבאה מציגה הגדרות נוספות של התגית <hr />:

מאפיין	משמעות	ערכים
align	יישור	left, center, right
noshade	ביטול הצללה	
size	עובי קו	pixels, %
width	אורך קו	pixels, %

בדוגמה שלפנינו מוצג הקובץ **Hr.html**. להלן הקוד בו משתמשים כדי ליישם את הדברים שלמדנו, ואת התוצאות בדפדפן ניתן לראות בתרשים 3.6.

```
<html>
  <head>
    <title>
      קווי רוחב
    </title>
  </head>
  <body>
    <hr size="12" width="150" align="left" />
    <hr size="9" width="105" align="right" />
    <hr size="6" width="73.5" noshade />
    <hr size="6" width="20%" align="center" />
    <hr size="10%" width="105" />
    <hr size="30%" width="50%" />
  </body>
</html>
```



תרשים 3.6: שליטה במראה הקווים במסמך נותנת לו מראה יצירתי יותר.

את **אורך הקו** (width) ואת **עובי הקו** (Size) אפשר לציין בנקודות (width="105") או כאחוזים (width="20%") ממידות החלון (אורך ורוחב) בו מוצג דף HTML.

כתובית, התגית <marquee>

Internet Explorer מאפשר לשלב תנועה בדפים שלך בדרכים שונות, אחת מהן היא הגדרת **כתוביות נעות** (scrolling marquee). מדובר בשורת טקסט הנגללת מעצמה לרוחב המסך. תוכל למשוך את תשומת לב הקורא למשפט מסוים או להברה, וכמובן שתוכל גם להגזים ולגרום לכך שכל המסך יתכסה בכתוביות נעות.

כדי ליצור כתובית נעה, הכנס את הטקסט בין תגיות התחום <marquee> ו-</marquee>. לדוגמה, התגיות הבאות יגרמו למחרוזת הטקסט "Watch me move!" לנוע מהגבול הימני של המסמך אל הגבול השמאלי ללא הפסק:

```
<marquee>Watch me move!</marquee>
```

מאפיין	משמעות	ערכים
height	גובה הכתובית	pixels, %
width	רוחב הכתובית	pixels, %
align	יישור	left, middle, top
bgcolor	צבע רקע	#rrggbb, blue, red
direction	כיוון תנועת הכתובית	left, right
behavior	התנהגות	scroll, slide, alternate
hspace	ריווח מלמעלה	pixels
vspace	ריווח מלמטה	pixels
scrollamount	מרווח בין כתוביות עוקבות	pixels
scrolldelay	השהיה	milliseconds (thousands)
loop	מספר הופעות	number, infinite

בקובץ **Marquee.html** תמצא מספר דוגמאות להפעלת כתובית נעה. בחן גם את קוד המקור:

```
<html>
  <head>
    <title>
      כתובית נעה
    </title>
  </head>
  <body>
    <marquee width="50%" align="top" direction="left">
    Here we go again</marquee> <br />
    <br />
     <marquee width="200" align="middle" direction="right">
    Here we go again</marquee> <br />
    <br />
     <marquee width="200" align="left" direction="right"
    behavior="alternate">Here we go again</marquee>
  </body>
</html>
```

שילוב תגיות

עד כה למדת מספר מועט של תגיות: <font>, <b>, <i>, <marquee> ועוד לפניך עשרות תגיות נוספות. אפשר לשלב בין התגיות, למשל, שילוב של התגית <b> עם התגית <font>. הקוד הבא, קובץ **TagPlusTag.html** מדגים שילוב של תגיות:

```
<html>
  <head>
    <title>
    </title>
  </head>
  <body>
    <font size="4" face="david">
        <b>David, Bold
            <br />
            <i>David, Bold, Italic
            </i>
        </b>
        <br />
        <i>David, Italic
        </i>
    </font>
  </body>
</html>
```

ל**תגיות משולבות** או ל**תגיות מקוננות** (Nested) יש כלל והוא "אחרון נפתח, ראשון נסגר" או אם תרצה "ראשון נפתח, אחרון נסגר". נדגים זאת על קטע הקוד הבא:

```
       <b>David, Bold
            <br />
            <i>David, Bold, Italic
            </i>
        </b>
```

התגית הראשונה היא (היא תהיה האחרונה שתיסגר). התגית אחריה היא <i>, ובמקרה של הדוגמה היא גם האחרונה, לכן היא תיסגר ראשונה </i>. רק עכשיו ניתן לסגור את התגית .

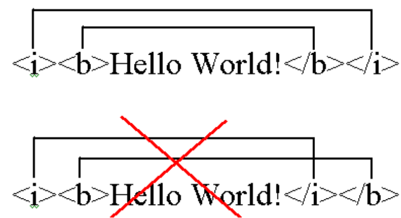
התגית <i> פועלת תחת ההשפעה של התגית <b> ולכן התגית <i> חייבת להיסגר בטווח הפעולה של התגית <b>.

התבונן בקטע הקוד הבא:

```
1  2  3  4
+           <body>
   +          <font size="4" face="david">
      +          <b>David, Bold <br />
         +          <i>David, Bold, Italic
         +          </i>
      +          </b><br />
      *          <i>David, Italic
      *          </i>
   +          </font>
+           </body>
```

ברמה הראשונה מוגדרת התגית <body>. ברמה השנייה, תחתיה, מוגדרת התגית <font>. מתחת לתגית <font> מוגדרות שתי תגיות ברמה שלוש: <b> ו- <i>. מתחת לתגית <b> ברמה שלוש מוגדרת, ברמה 4, התגית <i>.

ולסיכום, מה מותר ומה אסור:



שים לב, לפתיחה ולסגירה של כל תגית.

רווח לבן

HTML מתעלמת מ**רווחים לבנים** (White Spaces). אם תכתוב:

```
This is the
        Default font <br />
```

או

```
This is the Default font <br />
```

התוצאה תהיה זהה.

ועוד הפתעה, גם התוצאה של המשפט הבא:

```
This is                the            Default                    font <br />
```

תיראה אותו דבר כמו תוצאת שני המשפטים שמעליו.

אז בכל אופן, איך כותבים רווח שיוצג כמו רווח?

ובכן, יש להציג אותו בעזרת אוסף הסימנים ** ** - זהו הצירוף המורה לדפדפן להציג רווח. אפשר לכתוב רווחים זה אחר זה, כמה שתרצה.

ולכן, אם ברצונך להציג משפט שכזה:

```
This is                the            Default                    font <br />
```

ושכך הוא ייראה, יהיה עליך לכתוב אותו כך:

```
This is         the         
     Default             
               font <br />
```

סימנים מיוחדים

רווח היה אחד המקרים הפרטים של סימנים מיוחדים. סימנים מיוחדים אלה הם אותם סימנים שלא ניתן לכתוב אותם במסגרת דף HTML כי הם עשויים "להטעות" את הדפדפן, כאשר הוא מפרש את הדף. למשל, הסימנים < ו- >.

ניח, שתרצה להציג את הטקסט הבא a < c > b. אם תכתוב:

```
<font color="red">a <c> b</font>
```

מה שיוצג על המסך יהיה רק: a b כי <c> נחשב לתגית, אמנם זו תגית שאינה קיימת אך היא בכל זאת תגית.

פרט לסימנים >, < יש סימנים אחרים כמו ", & וגם © שצריך לייצגם בעזרת Character Entity.

להלן טבלה לסימנים המיוחדים השכיחים ביותר:

Character Entity	**תו**	**שם תו**
		Space
<	<	Less than
>	>	Greater than
&	&	Ampersand
"	"	Double quote
©	©	CopyRight
®	®	Registered trademark
'	‘	Apostrophe

לדוגמה ראה את קובץ **Hyphen.html**.

הערה

לעיתים יש צורך לרשום הערה כחלק מהקוד - הערה בשבילך, הערה שלא תוצג למשתמש אלא רק למי שמסתכל בקוד. זו יכולה להיות הערה, כמו למשל: "הקטע הבא מכיל את דברי הפתיחה" או "תאריך שינוי 21/3/2001" וכדומה.

בדף HTML רושמים הערה בין הסימנים <!-- לבין הסימנים --> באופן הבא:

```
<!-- written by Zohar Amihud 21/03/2001 -->
<!-- the next 5 lines are in italic
to emphasis the items in the list -->
```

הערות יכולות להירשם בכל מקום בדף, כל עוד הן מופיעות בין הסימנים הנדרשים. הערה יכולה להתפרס על יותר משורה אחת.

פרק 4

קישור

אחד מאבני היסוד של האינטרנט הוא הקישור (link). הקישורים הם אלה שהופכים את הרשת כל כך קלה לניווט ולשימוש.

הקישורים הם אבני היסוד של הרשת. בלחיצה קלה על קישורים אלה, המשתמש יכול לנוע באתר שלך, או לעבור לאתרים אחרים בכל מקום ברשת. הקישורים הם מה שעושה את הרשת לכלל-עולמית, וחשוב שתזכור תמיד שזה הדבר שהמבקרים באתר שלך מחפשים.

אך לפני שתלמד על קישורים יותר לעומק, עליך לדעת מהו URL.

מהו URL (Uniform Resource Locator)?

כבר סיכמנו כי קישורים הם הדרך לקשר בין קבצים ברשת האינטרנט. URL היא אותה כתובת ברשת אליה אנו מקשרים את הטקסט. המקום בו מופיעות כתובות אלו בדרך הברורה ביותר היא **שורת הכתובת** (address bar) של הדפדפן. ברוב הדפדפנים תוכל לראות כתובות אלו ב**שורת המצב** (status bar) כשתעביר את סמן העכבר מעל תמונה או טקסט שהוא קישור.

לכתובת URL שני מרכיבים עיקריים: הפרוטוקול והיעד.

הפרוטוקול מציין את סוג משאב הרשת מולו אתה עובד. הפרוטוקול השכיח ביותר הוא **//:http**, שתפקידו להעביר מסמכי HTML. בנוסף לו קיימים פרוטוקולים נוספים, כגון //:ftp (המציין שרת FTP להעברת קבצים).

היעד יכול להיות שם קובץ, תיקיה או מחשב ברשת. כתובת URL כמו **http://www.hod-ami.co.il/store/index.html** תעביר אותך ישירות לקובץ index.html, אך לא בכל המקרים תצטרך להשתמש בכתובת כל כך מפורטת. אם הכתובת היא **/http://www.hod-ami.co.il**, אז הדפדפן יתחבר לאתר ללא ציון שם קובץ מפורש אבל בפועל יפעיל את הדף שהוגדר כברירת מחדל לאתר זה.

קישור, התגית <a>

אם נתקלת בטקסט מואר עם קו תחתון או בצבע שונה משאר הטקסט, קרוב לוודאי שזהו **קישור**. תוכל לוודא זאת על ידי העברת סמן העכבר מעל הטקסט. אם הסמן משנה צורה מ- ל- משמע שזהו קישור. לידיעתך, יש גם קישורים שמעבר הסמן עליהם לא משנה את צורתם.

איך המחשב יודע לאן לעבור? לפי כתובת URL. תוכל למצוא קישור האומר "לחץ עלי כדי לקבל את מספרי הגרלת הלוטו". כל עוד הכתובת אליה מקושר הטקסט נכונה, סביר מאוד להניח שהדפדפן שלך יקרא לקובץ HTML בו מפורטים מספרי הגרלת הלוטו. כתובת URL מורה לדפדפן באיזה סוג קובץ מדובר (פרוטוקול) והיכן למצוא את הקובץ (יעד).

ניתן להגדיר קישור, בעזרת תגית הפתיחה <a> ותגית הסגירה . האלמנט המקושר יהיה זה שבא בין התגיות, היכול להיות טקסט וגם תמונה (על קישור תמונה ראה בפרק הבא). התגית <a> מכילה מיגוון רחב של מאפיינים המגדירים את יעד הקישור כמו גם את הגדרותיו.

להלן רשימה חלקית של מאפיינים לתגית **<a>**:

מאפיין	משמעות	ערכים
href	קישור לדף HTML או לעוגן במסמך	
name	שם של עוגן במסמך	
target	החלון בו ייפתח הקישור	_top, _blank, _self, _parent

קישור, התגית <a> והמאפיין href

קובץ **LinkTo.html** מדגים את השימוש בתגית **<a>** ליצירת קישור לאתר אינטרנט:

```
<body>
   <h1>Links to the most popular sites in Israel</h1>
   <a href="http://www.nana.co.il">Nana</a>
   <small>in the same window</small><br />
    <a href="http://www.walla.co.il" target="_blank">Walla</a>
   <small>in a new window</small>
</body>
```

כתובת URL עצמה חייבת להיות בין מרכאות וחייבת להופיע מייד לאחר הגדרת המאפיין **href**.

קישור ב-HTML נראה כך:

```
<a href="URL">טקסט הקישור נכנס כאן</a>
```

כלומר, אם אתה מעוניין לקשר את הטקסט "Nana" עם אתר בשם www.nana.co.il, עליך לכתוב את השורה הזו:

```
<a href="http://www.nana.co.il">Nana</a>
```

כאשר תפעיל את הקישור המופיע למעלה, האתר של Nana ייפתח באותו חלון בו היית. כדי להפעיל רק קישור שייראה בחלון חדש (נוסף), יהיה עליך להוסיף את המאפיין **target** באופן הבא:

```
<a href="http://www.walla.co.il" target="_blank">Walla</a>
```

אתה בוודאי שואל את עצמך כיצד ניתן לקשר בין דפי html הנמצאים באותו מחשב. זו פעולה שתבצע פעמים רבות במהלך עבודתך. בדרך כלל, תבנה לך רשת קטנה של מסמכים המוגדרת כאתר, ותשמור את כולם באותו מחשב-שרת. החשוב מכל הוא, שתרצה לקשר את דפי האתר שלך זה לזה. כשאתה מקשר בין דפי האתר שלך, אין צורך לציין את שם המחשב מכיון שכולם באותו המחשב. לדוגמה, אם הקובץ moreinfo.html נמצא באותה תיקיה ובאותו מחשב שנמצא הקובץ בו אתה יוצר את הקישור, כל שעליך לעשות כדי לקשר ביניהם הוא לכתוב את הקוד כך:

```
<a href="moreinfo.html">מידע נוסף</a>
```

קישור בתוך מסמך, התגית <a> והמאפיין name

אם אתה מעוניין ליצור קישור מחלק אחד של מסמך לחלק אחר (מסמכי HTML יכולים להיות ארוכים מאוד), כל שעליך לעשות הוא להוסיף את הסימן # לפני שם העוגן (anchor). הנה כך:

```
<a href="#phone">רשימת מספרי טלפון</a>
```

ברגע שתלחץ על קישור זה, יגלול הדפדפן את המסמך עד שיגיע ל**עוגן** (anchor) בשם "phone". כדי שיגיע לעוגן זה יש ליצור אותו כך:

```
<a name="phone">תוכל למצוא אותנו במספרים הבאים:</a>
```

שימוש כזה בעוגן המוצב בתוך מסמך HTML, מאפשר לנו להיות מאוד מדויקים ביצירת קישורים בתוך דפים וגם בין דפים. לדוגמה, נניח שיש לך מסמך מאוד ארוך, אליו אתה מעוניין ליצור קישור מדף אחר ברשת. בתוך הדף הארוך, בתחילת הקטע המעניין אותך לצורך הקישור, קיים עוגן. תוכל להוסיף לקישור שאתה יוצר, בנוסף לשם המחשב, תיקיה וקובץ, את שם העוגן במסמך.

```
<a href="http://www.myserver.com/hanger/moreinfo.html#phone">
רשימת הטלפונים של האתר שלי</a>
```

בקובץ **Links.html** תוכל למצוא דוגמה לקישורים בתוך המסמך. הפעל את הקישורים שבדף זה.

כעת אתה כבר יודע, פחות או יותר, כיצד פועלים קישורים וכתובות URL. הגיע הזמן לנסות ליצור מספר מסמכים ולקשר ביניהם. באותה הזדמנות צור גם כמה קישורים לאתרי אינטרנט מפורסמים, או לאתרים החביבים עליך.

אחד השימושים היותר נפוצים בקישור פנימי הוא הדילוג לתחילת המסמך. בדרך כלל קובעים את השורה:

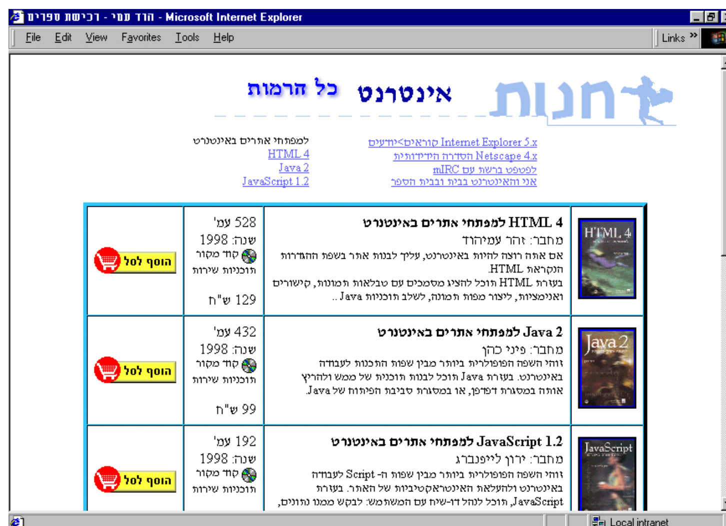
```
<a name="top"></a>
```

לאחר התגית <body>, אך זה לא הכרחי - אתה יכול לקבוע את "תחילת" המסמך היכן שנראה לך. תוכל להשתמש בקישור הפנימי כדי לאפשר למשתמש להגיע לראש הדף הנצפה בלחיצת עכבר, וכך זה נכתב:

```
<a href="#top">Going Up</a>
```

הנה דוגמה מעט יותר מורכבת. קובץ **Internet.html** מכיל קישורים פנימיים (שמות הספרים המופיעים בראש הדף) וקישורים חיצוניים לכתובות באינטרנט, והכל באותו דף.

נסה להפעיל את הקישורים (למרות שאינך יודע כיצד נבנה הדף):



תרשים 4.1

קישור לדואר אלקטרוני, התגית <a> והמאפיין href=mailto:

דואר אלקטרוני (E-Mail) הוא ללא ספק כלי מאוד פופולרי ברשת. הוא מאפשר לגולשים להתכתב ביניהם בכל נושא העולה על רוחם. כחלק מהעיצוב, מוסיפים מפתחי אתרים רבים, קישור לדואר האלקטרוני שלהם. קישור זה מאפשר למבקר באתר לשלוח הצעות והערות. אם באתר יש מידע רב ממקורות שונים, כדאי ליצור קישורים נוספים לדואר האלקטרוני של מקורות אלה.

הוספת קישור לדואר אלקטרוני במסמך HTML היא פעולה פשוטה, הדורשת רק כתובת E-Mail מדויקת. כתובת E-Mail בנויה משם המשתמש ושם domain, כשביניהם מפריד הסימן @ הנקרא at. שם המשתמש הוא, בדרך כלל, השם איתו אתה נכנס לרשת. שם ה-domain מורכב משם הארגון בעל השרת וסיומת.

דוגמה לכתובת דואר אלקטרוני:

info@hod-ami.co.il

אותה יש לקרוא כך: תיבת דואר בשם info בשרת בשם hod-ami.co.il.

כדי ליצור קישור אל כתובת E-Mail יש לכתוב כך:

```
<a href="mailto:info@hod-ami.co.il">Send me E-mail</a>
```

תרשים 4.2 מדגים כיצד נראה קישור מסוג :mailto.



תרשים 4.2: כשהמשתמש ילחץ על קישור E-Mail, ייפתח יישום הדואר האלקטרוני.

קישור לאתר FTP, התגית <a> והמאפיין href=ftp:

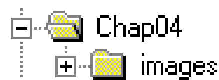
השימוש בפרוטוקול **FTP** (File Transfer Protocol) נעשה בעיקר כדי להעביר קבצים בין מחשב אחד לאחר. הגולשים ברשת נכנסים למחשב מרוחק, לרוב כ**אורחים** (guests), ומורידים מהם את הקבצים הנחוצים להם. **אורח** (guest) שנכנס למערכת FTP מזדהה בשם משתמש "anonymous", ובסיסמה שהיא כתובת הדואר האלקטרוני שלו.

כדי ליצור קישור לאתר FTP, עליך ליצור תגית קישור מתאימה. לדוגמה, קישור לאתר FTP של חברת Microsoft שכתובתו היא ftp.microsoft.com, ייראה כך :

```
<a href="ftp://ftp.microsoft.com/">Microsoft's FTP Site</a>
```

קישור יחסי

הקישור, כפי שציינו, הוא לדף HTML אחר הנמצא באותה תיקיה. תוכל להשתמש בהפניה יחסית לכתובת דף HTML, אם הוא נמצא בתיקיה אחרת. לדוגמה, מוגדרת תיקיה **Chap04** ותחתיה מוגדרת התיקיה **images**.



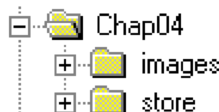
בתיקיה **Chap04** נמצא הקובץ **Internet.html** המפנה לדפים ו/או תמונות. אם למשל, הדף cdsmall.html נמצא באותה תיקיה שבה נמצא הקובץ **Internet.html** (כלומר תיקיה **Chap04**), אז הקישור ייכתב כך :

```
<a href="cdsmall.html"></a>
```

אבל אם הדף cdsmall.html נמצא בתיקיה **Chap04\images**, הקישור ייראה כך :

```
<a href="images/cdsmall.html"></a>
```

אם קיים מבנה התיקיות הבא :



וקובץ **Internet.html** נמצא בתיקיה store וקובץ cdsmall.gif נמצא בתיקיה images, אז הקישור צריך להיראות כך (הקישור נמצא בקובץ **Internet.html**) :

```
<a href="../images/cdsmall.gif"></a>
```

הסימן .. מורה לדפדפן לחפש את הקובץ cdsmall.gif החל מתיקיה הגבוהה ברמה אחת מהתיקיה הנוכחית. מכיון שהתיקיה store, בה נמצא קובץ **Internet.html**, מוגדרת תחת תיקיה **Chap04** - הנתיב יחל משם.

פרק 5

תמונות

תמונה שווה אלף מילים! יפוטרו המילים לאלתר...

ניתן ליצור דף אינטרנט ללא תמונות, והוא יעבוד. משתמשים מסוימים אפילו יודו לך על כך. אבל מתי ראית בפעם האחרונה פרסומת ללא תמונה? לוח מודעות? מגזין תעופה? אתר ברשת? אפילו בעיתונים יומיים הוסיפו כמות נכבדה של תמונות. כנראה שהעורך הבין שגם המוניטין היציב ביותר בעולם לא ימשוך את הקוראים לאורך זמן. לטוב או לרע, אנו חיים בעידן של תמונות, ואם אתה מעוניין למשוך לאתר שלך גולשים רבים לאורך זמן, רצוי שתעבוד על היכולת הגרפית שלך.

תמונות יכולות לשפר את מראה האתר:

- הן מפרידות את המסמך לחלקים ומקלות על הקריאה.
- הן מפרידות את נושאי המסמך השונים ומקלות על המבקר להתמצא במבנה האתר בדרך יעילה יותר.
- הן מציגות תוכן שלא ניתן לתאר במילים.
- הן מוסיפות צבע והומור לרשת.
- הן מפגינות את יצירתיות הכותב.



הערה

המילה תמונות או גרפיקה מתייחסות גם ל**איורים** (Clip-Arts), **סמלים** (Icons) וכדומה.

פורמטים גרפיים

לדפדפן Internet Explorer יש תמיכה מובנית במספר פורמטים גרפיים של קבצים, שהבולטים בהם:

GIF – Graphic Interchange Format

JPEG\JPG – Joint Photographic Expert Group

PNG – Portable Network Graphics

GIF

הפורמט הפופולרי ביותר באינטרנט. פורמט זה מצטיין בדחיסה וכך הופך קבצי תמונה גדולים לקבצים קטנים המתאימים לעבודה ברשת. פורמט זה מוגבל לעבודה עם 256 צבעים. ניתן לשמור קובץ **GIF** במבנה Interlaced. המשמעות היא, שהתמונה "תיבנה" תוך כדי טעינתה לדפדפן. הגולש יוכל לראות כי מתבצע תהליך של טעינת תמונה ובינתיים הוא יוכל לעיין בטקסט שבעמוד. תכונה נוספת לתמונות GIF היא ה**שקיפות** (transparency). מכיון שקטעי גרפיקה מוגבלים לצורה מלבנית, לא קיימת אפשרות לשמור עיגול או צורה אחרת שאינה מלבן. שקיפות, היא הדרך בה גורמים לצבע אחד להיעלם. כך, ניתן לשלב תמונה ברקע ובעיצוב הכללי שבדף. ולבסוף, ניתן בפורמט GIF ליצור **אנימציה** (animated GIF) על ידי שמירת מספר תמונות בקובץ אחד והצגתן בזו אחר זו במירווחי זמן שנקבעו מראש.

JPEG\JPG

פורמט **JPEG** (קוראים זאת jay-peg) לדחיסת תמונה. פורמט זה פותח במיוחד להקטנת קבצי תמונות מצולמות או גרפיקה המכילה הצללה מורכבת, אפקטים של תאורה ומיגוון רחב של גוונים וצבעים. המידע על הצבע לא נשמר במלואו וניתן להבחין בפגיעה באיכות התמונה הדחוסה ביחס לתמונה המקורית לפי איזורים מטושטשים או גסים. במשטחים בעלי גוון אחיד עדיף להשתמש ב-GIF ולא ב-JPEG.

PNG

פורמט **PNG** בעל שיעורי דחיסה טובים יותר, דבר שמקטין עוד יותר את גודל הקובץ בהשוואה ל-GIF. פורמט PNG אינו מוגבל ל-256 צבעים.

הוספת תמונה, התגית <img />

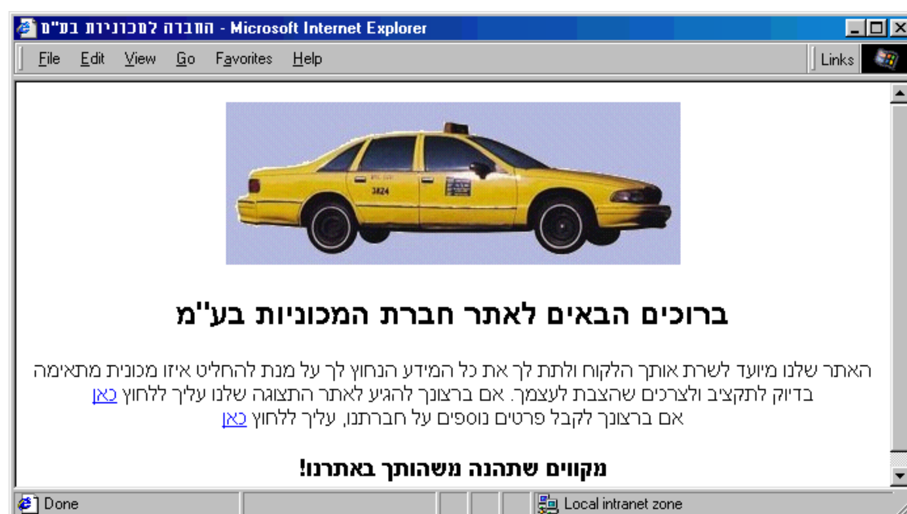
בעזרת עיצוב נכון, תמונה יכולה להיות שווה אלף מילים. אנו מצפים לתצוגה גרפית בכל דבר, וכך גם המבקרים באתר שלך באינטרנט מצפים לראות גרפיקה כחלק מהאתר.

להלן רשימת המאפיינים של התגית **<img />** :

מאפיין	משמעות	ערכים
src	שם התמונה כולל נתיב	URI (URL)
align	יישור הטקסט ביחס לתמונה	top, middle, bottom, left, right
alt	שם הכתובית כאשר הסמן יהיה על התמונה	text
border	ציור מסגרת לתמונה	pixels
height	גובה התמונה	pixels
width	רוחב התמונה	pixels
hspace	ריווח מהצדדים	pixels
vspace	ריווח מלמעלה ומלמטה	pixels

הוספת תמונה, התגית <img /> והמאפיין src

נתחיל עם תמונה. תרשים 5.1 מציג תמונה שנוספה למסמך HTML בעזרת תגית **<img />**. ראה קובץ **MyCar.html**.



תרשים 5.1 : פקודת HTML פשוטה מאפשרת לך להוסיף תמונות לתוך מסמכים.

כשאתה מעוניין להוסיף תמונה למסמך, אתה משתמש בכתובת URL שלה, ממש כמו ביצירת קישור. כתובת URL היא הכתובת המדויקת ברשת בה נמצא קובץ התמונה. התמונה יכולה להיות באותו מחשב בו נמצא האתר שלך, או במחשב מרוחק ברשת האינטרנט.

כדי ליצור קריאה לקובץ תמונה במסמך שלך, עליך להשתמש בתגית <img />. תגית זו מתנהגת כנקודת ציון שבה הדפדפן יציב את התמונה המבוקשת. התגית <img /> היא תגית ריקה במבנה הזה:

```
<img src="image_URL" alt="text" />
```

<img /> היא התגית להצגת גרפיקה בשפת HTML, והיא מופיעה בכל קריאה לגרפיקה.

המאפיין **src** (SouRCe) מתייחס למקור הקובץ ברשת. הכתובת עצמה מחליפה את הערך image_URL שבדוגמה הקודמת.

המאפיין **alt** מכיל את תוכן הכתובית, ברגע שסמן העכבר נמצא על הקישור.



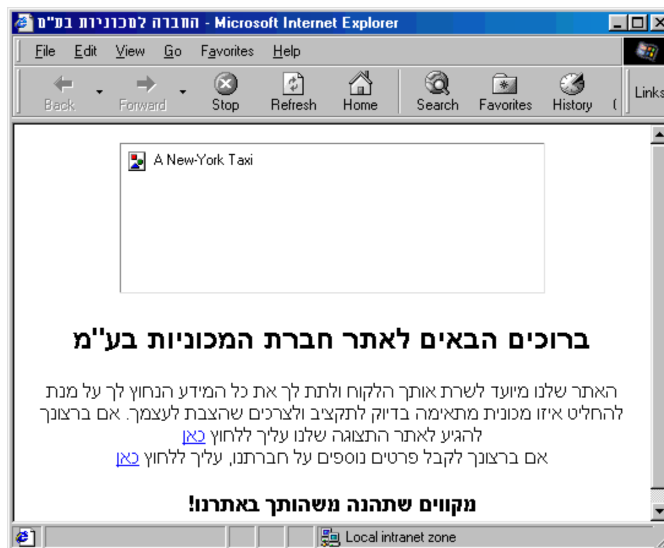
וכך נכתב הטקסט המלא לקישור בקובץ **MyCar.html**:

```
<img src="images\car.jpg" alt="A New-York Taxi" />
```

יש משתמשים המבטלים את אפשרות תצוגת הגרפיקה בדפדפן הגרפי שלהם כדי לזרז את משך טעינת דפי ה-Web למחשבים שלהם, אך גם אלה עדיין מעוניינים בגישה למידע שיש לאתר שלך להציע. מכיון שמטרת האתר היא העברת מידע למשתמשי הרשת, לא תרצה לאכזב עם דפדפנים טקסטואליים בלבד.

כיצד תאפשר למשתמשים אלה לדעת מה מסתתר מאחורי התמונה בה אתה משתמש? שפת HTML מאפשרת פתרון פשוט לסוגיה זו: מאפיין alt בתגית <img />. alt מגדיר משפט טקסט שיופיע במקום התמונה בדפדפנים טקסטואליים, או במקרה שאין אפשרות להציג תמונה. טקסט זה יוצג לרוב בתיבה המפרידה אותו מהטקסט.

תרשים 5.2 מדגים את תצוגת הטקסט בדפדפן, כשאין אפשרות לטעון את התמונה למסמך.



תרשים 5.2: חשוב שהטקסט יהיה קצר למדי ושיתאר את התמונה לאלה שלא רואים אותה.

כתובת URL של תמונה יכולה להיות נתיב מלא הכולל את שם המחשב, תיקיה ושם קובץ (http://www.hod-ami.co.il/images/books.jpg) או כתובת יחסית. במקרה זה, תתייחס למיקום הקובץ ביחס לתיקיית המסמך.

דוגמת הקוד הבאה - **ImgDemo.html**, מדגימה הוספת קובץ תמונה בתוך טקסט:

```
<p>
    I just thought you might be interested in seeing this graphic
    I have created for myself in PhotoShop.
    <img src="images/home.gif" alt="Home Banner" />
    I was actually a bit surprized at how
     :
     :
</p>
```

שים לב שהתגית <img /> משתמשת כאן בכתובת יחסית. באותה מידה התגית יכולה היתה להשתמש בכתובת מלאה:

```
<img src="http://www.mycom.com/images/home.gif" />
```

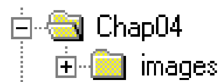
בנתיב מלא רצוי להשתמש כשהקובץ המיועד נמצא במחשב אחר באינטרנט. אם הקובץ היה נמצא בתיקיית משנה, בשם graphics, שהיא תת-תיקיה של התיקיה בה נמצא דף HTML בו יש קריאה לתמונה, ניתן היה להשתמש בכתובת יחסית כזו:

```
<img src="graphics/silvia.gif" />
```

כתובת זו תקינה, אם הקובץ silvia.gif נמצא בתיקיית המשנה graphics.

מיקום יחסי

אם במאפיין src נמצא רק שם של קובץ, ללא נתיב, ברירת המחדל של הדפדפן היא לחפש בתיקיה בה נמצא הקובץ (HTML) שקרא לתמונה. ניתן להשתמש בהפניה יחסית לתמונה (ממש כפי שנעשה עם קישורים) אם היא נמצאת בתיקיה אחרת. לדוגמה, מוגדרת תיקיה **Chap04** ותחתיה מוגדרת התיקיה images.



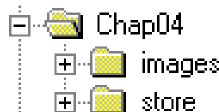
בתיקיה **Chap04** נמצא הקובץ **Internet.html** המפנה לדפים ו/או תמונות. אם למשל, התמונה cdsmall.gif נמצאת באותה תיקיה שבה נמצא הקובץ **Internet.html** (כלומר תיקיה **Chap04**), אז התגית <img /> תיראה כך:

```
<img src="cdsmall.gif" />
```

אבל אם הקובץ cdsmall.gif נמצא בתיקיה **chap04\images**, אז התגית <img /> תיראה כך:

```
<img src="images/cdsmall.gif" />
```

אם קיים מבנה התיקיות הבא:



ואם קובץ **Internet.html** נמצא בתיקיה store וקובץ cdsmall.gif נמצא בתיקיה images, אז התגית <img /> תיראה כך:

```
<img src="../images/cdsmall.gif" />
```

הסימן .. מורה לדפדפן לחפש את הקובץ cdsmall.gif החל מהתיקיה הנמצאת ברמה אחת גבוהה יותר מהתיקיה הנוכחית. מכיון שהתיקיה store, בה נמצא קובץ **Internet.html**, מוגדרת תחת תיקיה **Chap04** - יחל הנתיב משם.

טקסט ותמונה, התגית <img /> והמאפיין align

הדפדפנים עצמם אינם יוצאים מגדרם כדי לעזור לטקסט ולגרפיקה לחלוק שטחים במסמכי HTML.

אך יש מה לעשות בנידון! התגית **<img />** מאפשרת גם מאפיינים מסוג **align**. בעזרת מאפיינים אלה ניתן לקבוע את יחסי הטקסט והתמונה. אפשרות זו משפיעה על הדרך בה יופיע טקסט לצד גרפיקה, באותו משפט.

במאפיין align נשתמש כך:

```
<img align="value" src="image_URL" />
```

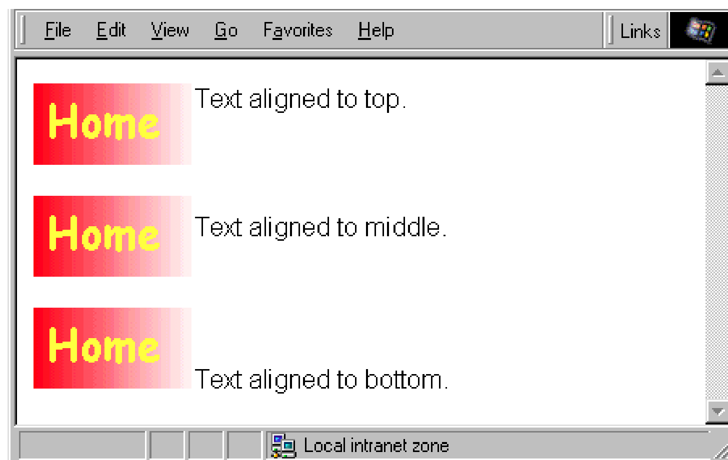
הערכים האפשריים למאפיין **align** הם:

ערך	השפעה על הטקסט
middle	מציב את הטקסט במרכז התמונה
bottom	מציב את הטקסט בתחתית התמונה
top	מציב את הטקסט בחלק העליון של התמונה
left	מציג את התמונה בצד שמאל של החלון
right	מציג את התמונה בצד ימין של החלון

ערך ברירת המחדל של המאפיין align בתגית <img /> הוא bottom, כך שלא תצטרך לציין דבר כדי להשתמש בערך זה. כשתשתמש באחד מערכי ברירת המחדל של התגית <img />, הדפדפן ישאיר חלל ריק סביב הטקסט בשורה, והטקסט יגלוש לשורה הבאה מתחת לתמונה. קובץ **AlignDemo.html** מדגים שימוש בערכי ברירת המחדל של תגית <img /> בקובד המסמך שלך (קוד חלקי מתוך המסמך):

```
<img src="images\home.gif" align="TOP" />Text aligned to top.
<img src="images\home.gif" align="MIDDLE" />Text aligned to middle.
<img src="images\home.gif" align="BOTTOM" />Text aligned to bottom.
```

תרשים 5.3 מדגים את תוצאות הקוד.



תרשים 5.3: רק טקסט בשורה הראשונה יהיה בגובה שצוין במאפיין align. שאר הטקסט יופיע מתחת לתמונה.

תמונות צפות, התגית <img /> והמאפיינים <vspace> ו-<hspace>

תמונות צפות אינן קשורות לשורה אחת בלבד של טקסט. תמונות אלו נמצאות בצמוד לשולי המסמך או במרכזו, והטקסט זורם לצידן באופן מושלם.

הערכים של המאפיין align לתמונה צפה הם left ו-right. מאפיין זה מציין לאיזה צד של שולי הדף תוצמד התמונה. קובץ **SpaceNO.html** מדגים כמה קל להשתמש בהגדרות אלו.

```
<img src="images\smallcar.jpg" align="left" />
```

```
<img src="images\small2cars.jpg" align="right" />
```

תרשים 5.4 מציג כיצד הדפדפן מפרש את מאפיין התגית <img /> ומציג תמונות צפות.



תרשים 5.4 : תמונות צפות מנותקות מהטקסט ולכן מאפשרות לו לזרום סביבן.

קיימים שני מאפיינים נוספים כדי לשלוט ברווח שבין תמונה צפה, טקסט ושולי המסמך - vspace ו-hspace. המאפיין vspace מגדיר את הרווח בין התמונה למה שמעליה ומתחתיה (במאונך לה), ואילו המאפיין hspace מגדיר את הרווח בין התמונה למה שמימינה ומשמאלה (במאוזן לה). קובץ **SpaceYES.html** מציג מאפיינים אלה בפעולה.

```
<img src="images\smallcar.jpg" hspace="10" vspace="10" align="left" />
```



בעברית פשוטה!

המשמעות של **vspace** היא **רווח אנכי** (Vertical) ואילו המשמעות של **hspace** היא **רווח אופקי** (Horizontal). כך שהמשפט:

```
<img src="aa.gif" align="left" hspace="10" />
```

מגדיר תמונה שצפה במקביל לשולי המסמך השמאליים, והרווח בינה לבין השוליים והטקסט יהיה 10 פיקסלים.

אם תשווה את תרשימים 5.5 ו- 5.6, תוכל לראות את ההבדל בין תמונות בלי/עם הגדרות המאפיינים **hspace** ו-**vspace**.



תרשים 5.5: הגדרות hspace ו-vspace משפיעות על שני צידי התמונה. אין אפשרות להשפיע רק על צד אחד בלבד.

גודל תמונה, התגית <img /> והמאפיינים width ו-height

שני מאפיינים חשובים לתגית <img />: **height** ו-**width**. מאפיינים אלה מאפשרים קריאה מהירה מעט יותר של המסמך. כיצד?

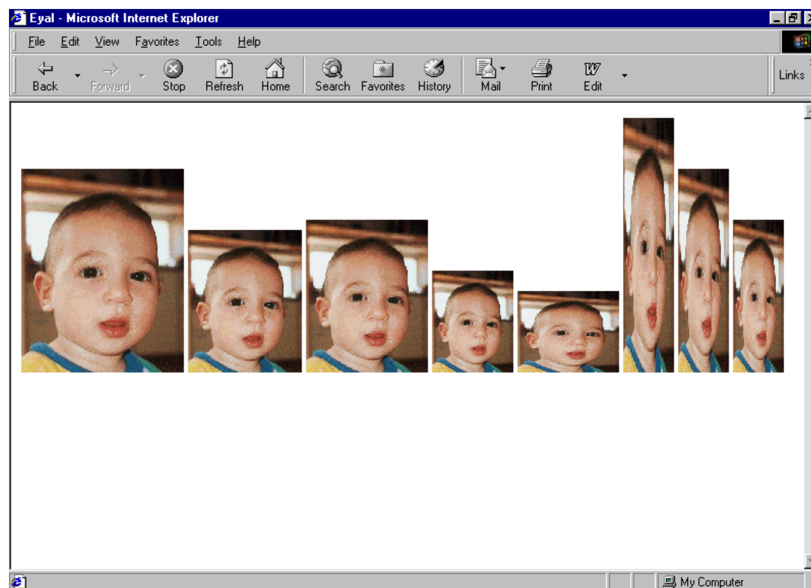
height ו-width הם מאפיינים המסייעים בפתרון אחת הבעיות המציקות ביותר, הגוזלות זמן ברשת. כשגולש מבקר באתר שלך, הדפדפן שלו ימתין עד שכל התמונות ייטענו לזיכרון, ורק אז יציג את המסמך, כיון שהדפדפן מבקש לדעת כמה חלל יש

להקציב במסמך לטובת הגרפיקה. המאפיינים height ו-width מפחיתים את משך זמן ההמתנה - הם מודיעים מראש לדפדפן את הגודל המדויק של הגרפיקה במסמך.

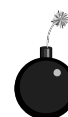
כשהדפדפן יודע מראש את גודל הגרפיקה במסמך, הוא יכול לבנות את המבנה הבסיסי עוד לפני שטען לזיכרון את הגרפיקה עצמה, שהיא המרכיב הכבד ביותר במסמך מבחינת זיכרון. כך, המשתמש יכול להתרשם מהמסמך, ולעיתים קרובות לעבור למסמך אחר עוד לפני שהדפדפן הספיק לטעון את כל הגרפיקה.

מקרה שכיח הוא, שהתמונה מוצגת בצורה מעוותת בגלל חוסר התאמה בין הערכים של המאפיינים height ו-width יחסית לגודל המקורי של התמונה. לדוגמה, התמונה eyal.gif היא בגודל 160x200, כלומר רוחב של 160pixels וגובה של 200pixels. תוכל להציג תמונה זו יותר גדולה ויותר קטנה בעזרת המאפיינים width ו-height, אבל יהיה עליך לשמור על פרופורציה נכונה. למשל, אם קבעת שרוחב התמונה צריך להיות 100 ולא 160, אז הגובה צריך להיות 125. ההסבר: 100 מתייחס ל-160 (הרוחב המקורי) כמו x ל-200 (הגובה המקורי). לכן, הערך של x הוא 125. דוגמה נוספת, אם אתה רוצה להקטין את התמונה ב-20%, אז הרוחב יהיה 112 (160*0.7=112) והגובה יהיה 140 (200*0.7=140). קובץ **ImageSizes.html** מדגים את השימוש במאפיינים height ו-width היכולים לפעמים לשבש את התמונה. רק ארבע התמונות משמאל מוצגות נכון מבחינת היחס בין הגובה לרוחב:

```
<html>
  <head>
    <title>
      Eyal
    </title>
  </head>
  <body>
    <img src="images\eyal.gif" />
    <img src="images\eyal.gif" width="112" height="140" />
    <img src="images\eyal.gif" width="120" height="150" />
    <img src="images\eyal.gif" width="80" height="100" />
    <img src="images\eyal.gif" width="100" height="80" />
    <img src="images\eyal.gif" width="50" height="250" />
    <img src="images\eyal.gif" width="50" height="200" />
    <img src="images\eyal.gif" width="50" height="150" />
  </body>
</html>
```



תרשים 5.6



אזהרה

למרות שהמאפיינים **height** ו-**width** מסוגלים לשנות את גודל תצוגת התמונה, הם **אינם** משנים את גודלה הפיסי. כלומר, המשתמש צריך להמתין את אותו משך הזמן לטעינת התמונה, בין אם הוא יראה אותה בגודל טבעי או בגודל אחר.

קישור תמונה

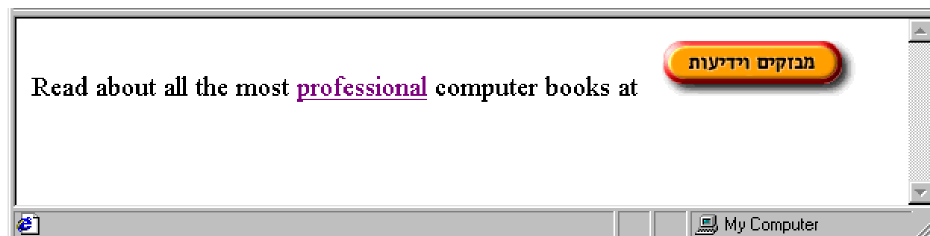
כעת, כשאנו יודעים כיצד ליצור קישורים וכיצד להוסיף תמונות לתוך מסמכי HTML, למה לא לשלב בין שתי הפעולות וליצור משהו חדש? קוד HTML לפעולה זו יהיה:

```
<a href="URL"><img src="imagename.gif" alt="text" /></a>
```

כפי שניתן לראות, במקום ליצור קישור בעזרת טקסט בדף ה-Web, אנו יוצרים קישור בתמונה המוגדרת בעזרת תגית <img />. התמונה מופיעה על המסך והמשתמש לוחץ עליה במקום על טקסט.

ראה קובץ **News.html**.

```
<html>
 <head>
  <title>
   Hot News
  </title>
 </head>
 <body>
  <font size="4">Read about all the most
  <a href="http://www.hod-ami.co.il">professional</a> computer books at
  <a href="http://www.hod-ami.co.il">
  <img src="images\news.gif" border="0" alt="Hod-Ami Computer books" /></a>
  </font>
 </body>
</html>
```



תרשים 5.7: קישור לטקסט וקישור לתמונה.

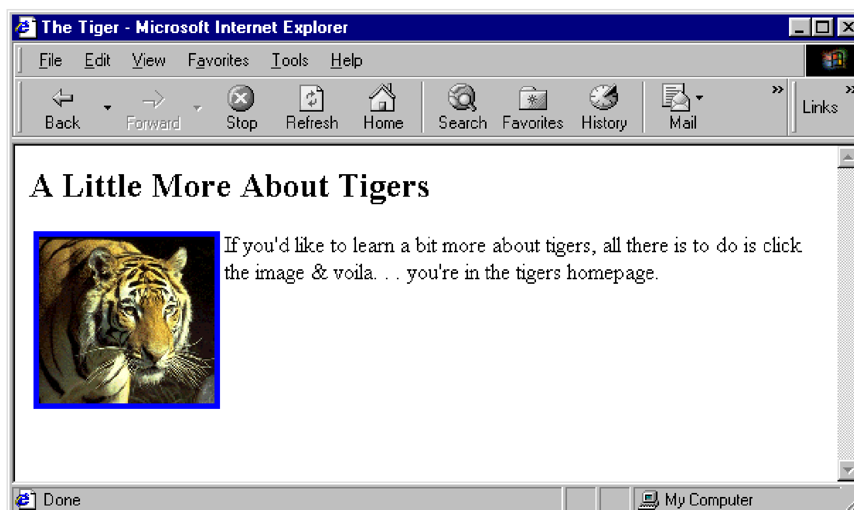
שים לב למאפיין **border** לתגית **<a>**. מאפיין זה יוסבר מייד בהמשך.

גבול לתמונה, התגית <img /> והמאפיין border

תוספת זו שימושית במיוחד כשאתה יוצר תמונה מקושרת. המאפיין **border** קובע את עובי המסגרת מסביב לתמונה. ערך ברירת המחדל של המאפיין border הוא 1. אם תשנה זאת לערך גבוה יותר, תהיה המסגרת מסביב לתמונה עבה יותר. אם תגדיר ערך 0, תיעלם המסגרת המעידה כי כאן קיים קישור. ראה קובץ **TheTiger.html**.

כדי להשתמש במאפיין **border**, יש לכתוב את התגית **<img />** כך:

```
<img src="images\tiger.gif" border="4" align="left" alt="Tiger" />
```



תרשים 5.8: לפניך קישור גרפי. כדי לעבור לדף הבא עליך ללחוץ עליו.

צבע המסגרת סביב התמונה תלוי בהקשר. במילים אחרות, אם התמונה היא גם קישור, המסגרת תהיה בצבע כחול (זהו צבע ברירת המחדל של קישור). אם התמונה אינה מקושרת, הצבע יהיה שחור.

שאלה ותשובה!

אם אני מבטל את המסגרת, איך המבקרים באתר יידעו שזה קישור?

במראה התמונה אכן תהיה רגילה, מה שעלול לבלבל את המבקר באתר. כשאתה משתמש בערך "border="0, חשוב שתוסיף לצד התמונה גם טקסט שיודיע למשתמש שהתמונה משמשת גם כקישור.

עוד על גרפיקה תוכל לקרוא בפרק הבא.

פרק 6

צבע

תקשורת חזותית היא מהנה וגם מהווה אתגר. גרפיקה (תמונות, איורים, סמלים) יכולה לעזור להתגבר על בעיות תקשורת בינלאומיות, אך עדיין לא מוציאה את הטקסט לגמרי אל מחוץ לתמונה.

טפט, התגית <body> והמאפיין background

בוודאי חיכית שנציג נושא זה כבר זמן רב. נראה כי מפתחים רבים נהנים מאוד להדביק טפט כלשהו כרקע לאתר שלהם, שכמובן נעשה בעזרת גרפיקה ולא בעזרת דבק. פעולה זו, ללא ספק, מוסיפה אופי מיוחד לאתר, אם זאת כוונתך. ראה קובץ **BackGround.html**.

הדרך לעשות פעולה זו היא פשוטה מאוד:

```
<body background="images\paper.gif">
```

בוודאי מעניין אותך לדעת מה גודל תמונת הרקע (קובץ paper.gif)? ובכן, גודלה האמיתי של התמונה הוא 108 x 212.

תוכל לנצל את יכולת "ההכפלה" של הדפדפן כדי לקחת תמונה כמו זו (edge.gif)



וליצור איתה רקע מעניין כמו זה הנראה בקובץ **Edge.html**.

צבע

אם אתה באמת רוצה להתפרע ולתפוס שליטה על כמה שיותר מרכיבים במסמכים שלך, הגיע הזמן שנסיר את הלוט מעל חלק מפקודות הצבע. את הצבע אפשר לשלב בהרבה תגיות שאת חלקן כבר למדת: <font>, <body>, <marquee>, <basefont />.

את הצבע ניתן להגדיר בדף HTML בשתי צורות:

1. בשם, לדוגמה: blue, green, yellow וכדומה.
2. בפורמט **RGB**.

פורמט RGB הינו הגדרת צבע על ידי ערבוב הצבעים: אדום (Red), ירוק (Green) וכחול (Blue). לפי מודל צבעי RGB נוכל לערבב יחסים שונים של שלושת הצבעים (אדום, ירוק, כחול) כדי לקבל כל צבע החל משחור (וזה ברור) ועד לבן (וזה גם ברור) וכולל ורוד, ירוק ומאות אלפי גוונים נוספים. כל פרמטר, מתוך השלושה, יכול לקבל ערך מ-0 ועד 255. המשמעות היא, שמספר הצירופים האפשריים הוא 16.7 מיליון צבעים.

למשל, הצירוף של אדום, ירוק, כחול בערכים 255, 255, 255 בהתאמה ייתן צבע לבן, כמו שצירוף 0, 0, 0 יציג צבע שחור.

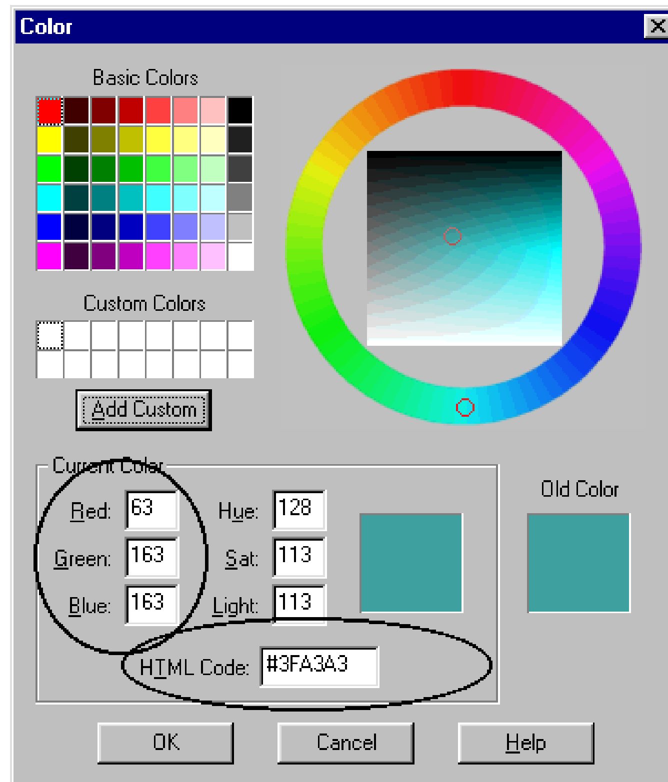
הנה לפניך צירופים אחרים:

צבע/ RGB	כחול (Blue)	ירוק (Green)	אדום (Red)
צהוב	0	255	255
סגול	165	40	165
ירוק כהה	163	163	63
ורוד	255	0	255

הבעיה בצבעי RGB היא, שלא ניתן להגדיר לדפדפן את הצבע בעזרת שלושת הערכים הדצימליים של RGB אלא חייבים לציין זאת בקוד הקסדצימלי.

בשיטת הספירה הדצימלית בה אנו עושים שימוש יום יום, קיימות הספרות 0, 1, 2, 3, 4, 5, 6, 7, 8 ו-9. בשיטת ספירה הקסדצימלית קיימות הספרות: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E ו-F. הערכים הדצימליים יתורגמו לערכים הקסדצימליים.

בתרשים 6.1 תוכל לראות את הגדרת הצבע בפורמט RGB עם ערכים עשרוניים (דצימליים) וגם את ההגדרה עבור דפי HTML.



תרשים 6.1: כך נראה חלון בחירת צבע בתוכנת Paint Shop Pro.

אם כך, הגדרת הצבע R=63, G=163, B=163 בדף HTML תירשם #3FA3A3.

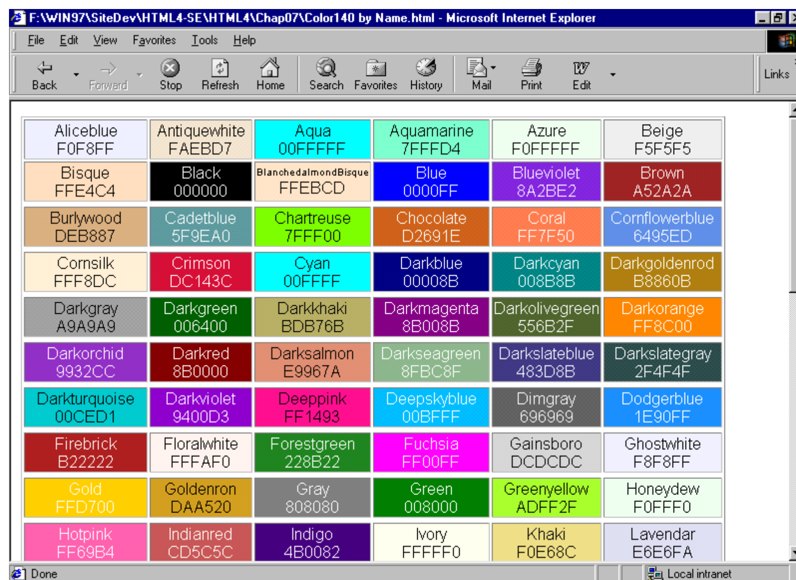
כדי להקל עליך את העבודה, תוכל להשתמש בקובץ **Color140byName.html** כדי לראות הגדרה של 140 צבעים בשמם (אותם תוכל לציין בדף שתבנה) או בפורמט RGB. כדי להשתמש בערכים שבטבלה, סמן את הטקסט והעתק (Ctrl+C) את ערכו.

תוכל להשתמש בשם הצבע כמו Red, Yellow וגם Deeppink, Blueviolet, Dimgray או בקוד ההקסצימלי שלו: #FF0000, #FFFF00, #FF1493, #8A2BE2 ו- #696969 בהתאמה.



הערה

אל תשכח לשים את הסימן **#** לפני המספר בפורמט ההקסצימלי.



תרשים 6.2 : פלטת הצבעים. הפעל את הקובץ **Color140byName.html**.

לרשותך מיליוני צבעים נוספים אותם תוכל לשלב בדף HTML, אבל יהיה עליך לזכור שהדפדפן יכול להציג רק 216 צבעים. למעשה, הדפדפן יכול להציג 256 צבעים, אבל רק 216 מהם בטוחים להצגה על המסך. בעגה המקצועית אוסף הצבעים הבטוחים להצגה נקרא **Safety Pallete**. יתר 40 הצבעים עשויים להשתנות בין הפלטפורמות השונות (Mac ו-PC).

Color140byName.html	140 צבעים המסודרים בסדר אלפביתי
Color140RGBorder.html	140 צבעים מסודרים לפי מספר הצבע בפורמט RGB

צביעת רקע, התגית <body> והמאפיין bgcolor

נתחיל בצביעת הרקע. תוך שימוש במאפיין **bgcolor** של התגית **<body>**, תוכל ליצור צבע רקע למסמך בעזרת ציון מספר הקסדצימלי בן שש ספרות :

```
<body bgcolor="#rrggbb">
גוף המסמך
</body>
```

הערך #rrggbb מציין את הערך הדו-ספרתי של הצבעים אדום, ירוק וכחול, בהתאמה, בצבע הרקע של המסמך. לדוגמה, אם אתה מעוניין לשנות את רקע האתר שלך לשחור השתמש בקוד :

```
<body bgcolor="#000000">
```

יכולת גם לכתוב את הקוד כך :

```
<body bgcolor="black">
```

ולהגיע לאותה תוצאה בדיוק - רקע שחור.

אתה לא צריך לזכור את כל הקודים ההקסדצימליים ו/או את שמות הצבעים בעל-פה. תוכנות גרפיות כמו **PSP** מציגות לך את קוד הצבע לגבי כל צבע שתבחר, וכן תוכל להיעזר בקבצים : **Color140byName.html** או בקובץ **Color140RGBorder.html**.

בדומה לשחור, ניתן לציין את הערכים של הצבעים הבאים כך :

צבע		**RGB**
תכלת	aqua	00FFFF
שחור	black	000000
כחול	blue	0000FF
אדום כהה	fuchsia	FF00FF
אפור	gray	808080
ירוק	green	008000
ירוק בהיר	lightgreen	00FF00
ירוק זרחני	lime	00FF00
חום	maroon	800000
כחול כהה	navy	000080
זית	olive	808000
סגול	purple	800080
אדום	red	FF0000
כסף	silver	C0C0C0
ירוק כהה	teal	008080
לבן	white	FFFFFF
צהוב	yellow	FFFF00

ספרי ההדרכה בהוצאת **הוד-עמי** על תוכנת **paint shop pro** עוסקים בהרחבה בנושא גרפיקה וצבע.

הספר **עיצוב ממשק באינטרנט** בהוצאת **הוד-עמי**, יספק לך מידע רב בנוגע לשילוב הצבעים באתר, באופן שיתמוך גם במראה וגם ביכולת הניווט של המשתמש באתר.

צביעת טקסט, התגית <body> והמאפיין text

לאחר ששינית את צבע הרקע, ייתכן שתצטרך לשנות, אם תרצה, גם את צבע הטקסט במסמך. ברירת המחדל לצבע טקסט ברוב הדפדפנים היא שחור (black, #000000), מלבד קישורים.

טיפ

קובץ **Color140byName.html** מאפשר לך לבחור צבעים עבור הרקע, הטקסט והקישורים.

כדי לשנות את צבע הטקסט הכללי במסמך, השתמש בתכונה **text** של התגית **<body>** כך:

```
<body text="#rrggbb">
גוף המסמך
</body>
```

בקוד זה, #rrggbb מייצג סדרה של מספרים הקסדצימליים, המייצגים שילוב בין אדום, ירוק וכחול. למשל, הקוד הבא מציג את הטקסט בצבע כחול בהיר:

```
<body text="Lightskyblue">
```

שזה בדיוק כמו לכתוב:

```
<body text="#87CEFA">
```

צבע קישור, התגית <body> והמאפיין link

צבעי ברירת המחדל של קישור הם כחול (#0000FF) לקישור חדש, וסגול (#800080) לקישור שביקרת בו בעבר. ניתן לשנות את צבע הקישורים כרצונך. קיימים שני סוגי קישורים:

1. **link** - קישור.
2. **vlink** - קישור שביקרת בו.

כדי לקבוע את ערכם של אלה, עליך להשתמש בקוד:

```
<body link="#rrggbb" vlink="#rrggbb">
גוף המסמך
</body>
```

שוב, rrggbb# מייצג את השילוב בין אדום, ירוק וכחול. ברירות המחדל למאפיינים אלה הן: **כחול** (0000FF#) עבור **link**, **סגול** (800080#) עבור **vlink**. ייתכן שצבעים אלה שונים, אם המשתמש הגדיר אותם אחרת.

לדוגמה, כדי לקבוע צבע **ירוק זית** (Olive או 808000#) לקישור וצבע **אדום** (red או FF000#) לקישור שביקרת בו, כתוב כך את תגית <body>:

```
<body link="#808000" vlink="red">
```

בנקודה זו של שינוי צבע הקישורים יש לשים לב. צבע ברירת המחדל של קישור הוא כחול, כך זה במרבית האתרים בעולם וכך רגילים כל הגולשים. אמנם אתה יכול לצבוע את הקישור באיזה צבע שתחפוץ, אבל האם הגולשים באתר שלך יזהו שזה קישור? על כך ועל שימושים אחרים בצבע קרא בספר **עיצוב ממשק באינטרנט** בהוצאת **הוד-עמי**.

צביעת טקסט, התגית והמאפיין color

עד כה, כל פקודות הצבע שראינו היו כלליות מאוד. למעשה, ניתן לשנות את צבעם של מילים בודדות, משפטים או קטעים בודדים, מבלי להשפיע על שאר הטקסט. כל שעליך לעשות הוא, לתחום את הטקסט אותו ברצונך לעצב בין התגיות **** ו-****:

```
<font color="red">this text is red</font>
```

ההגדרה של צבע האות יכולה להיות הקסדצימלית (FF0000#), או במילים פשוטות וברורות יותר (red). אתה תחליט מה מתאים לך יותר.

פרק 7

טבלאות

טבלאות HTML מציגות סוגים שונים של נתונים בדרך נוחה ומסודרת לקריאה. ייחודן הנוסף של הטבלאות, הוא בכך שהן מאפשרות לך לעמד את המסמכים שלך כמעט כמו ב-Word.

יצירת טבלה, התגית <table>

טבלה נמצאת בין תגיות התחום **<table>** ו-**</table>**. הדרך הקלה ביותר לחשוב על טבלה כאוסף של תאים בודדים המסודרים בעמודות ובשורות. תא הוא המפגש של עמודה ושורה.

> **בעברית פשוטה!**
>
> **שורות, עמודות ותאים** הם המרכיבים הבסיסיים של טבלה, אותם אתה בוודאי מכיר מתוכנות Office כדוגמת Word, Excel או Access. שורות מוצגות לרוחב המסך ועמודות לאורכו. בכל מפגש בין שורה לעמודה, נוצר תא. חשבון פשוט: אם בטבלה עשר שורות ושלוש עמודות - בטבלה יש ... שלושים תאים.

הגדרת שורה בטבלה, התגית <tr>

את הטבלה אתה בונה לפי שורות. כלומר, אם ברצונך להגדיר תאים בשורה מסוימת, ראשית עליך להגדיר את השורה. זה לא מסובך, וכל שעליך לעשות הוא לתחום את כל התאים בשורה מסוימת בין התגיות **<tr>** ו-**</tr>**.

הגדרת תא בטבלה, התגית <td>

את השורה אתה בונה מתא/ים. כלומר, אם ברצונך להגדיר תאים בשורה מסוימת, ראשית עליך להגדיר את השורה (את זה עושים עם התגית <tr>), ואחר כך להגדיר כל תא ותא בנפרד. זה לא מסובך, וכל שעליך לעשות הוא לתחום את הערך של תא (טקסט, תמונה, עם/בלי קישור) בין התגיות **<td>** ו-**</td>**.

קובץ **Table01.html** מדגים טבלה פשוטה:

```
<table>
    <tr>
      <td>Word</td>
      <td>Excel</td>
      <td>Access</td>
    </tr>
    <tr>
      <td>WinZip</td>
      <td>WinAmp</td>
      <td>WinGate</td>
    </tr>
  </table>
 </body>
</html>
```



תרשים 7.1

התגית **<table>** פותחת את הגדרת הטבלה. שים לב שלא ניתן לראות את קווי הטבלה. כיצד להציג את קווי הטבלה תלמד בהמשך.

התגית **<tr>** מתחילה את הגדרת השורה הראשונה.

אחרי התגית **<td>** יבוא הערך של התא, ובסיומו התגית </td>.

השורה הראשונה בטבלה מוגדרת באמצעות התגית <tr>, ובה שלושה תאים. את זה יודעת הטבלה, תוך כדי בנייה, מכיון ששלוש פעמים מתחילה התגית <td> ושלוש פעמים היא נסגרת </td>.

בסיום מתן הערכים לתאים, בין התגית <td> לתגית </td>, יש צורך לסגור את הגדרת השורה על ידי התגית </tr>.

באופן דומה מוגדרת השורה הבאה בטבלה.

בגמר הגדרת השורה השנייה נסגרת הטבלה על ידי התגית </table>.

מאפייני התגית <table>

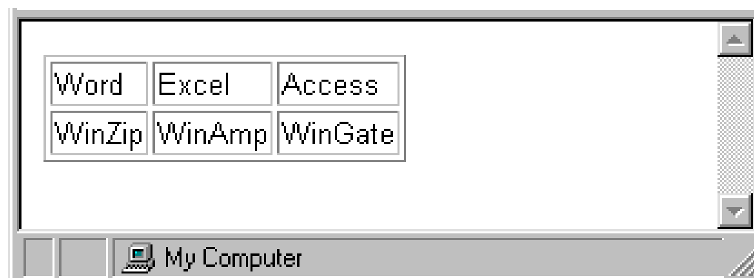
הטבלה שהתקבלה זקוקה למעט קוסמטיקה, ולכן זה הזמן המתאים ללמוד על המאפיינים של התגית **<table>**.

מאפיין	משמעות	ערכים
align	יישור הטבלה יחסית למסך	left, center, right, justify
bgcolor	צבע רקע לטבלה	blue, green.... #rrggbb
border	גבול הטבלה	pixels
bordercolor	צבע המסגרת	blue, green.... #rrggbb
cellpadding	מרווח בין תוכן התא לגבולותיו	pixels, %
cellspacing	מרווח בין תאים	pixels, %
frame	איזה גבולות ייראו מסביב לטבלה	void, above, below, hsides, lhs, rhs, vsides, box, border
width	רוחב כל הטבלה	pixels, %

גבולות טבלה, התגית <table> והמאפיין border

קובץ **Table02.html** מראה טבלה עם גבול, כאשר השוני לעומת קובץ **Table01.html** הוא בהוספת המאפיין border לתגית <table>, באופן הבא:

```
<table border="1">
```



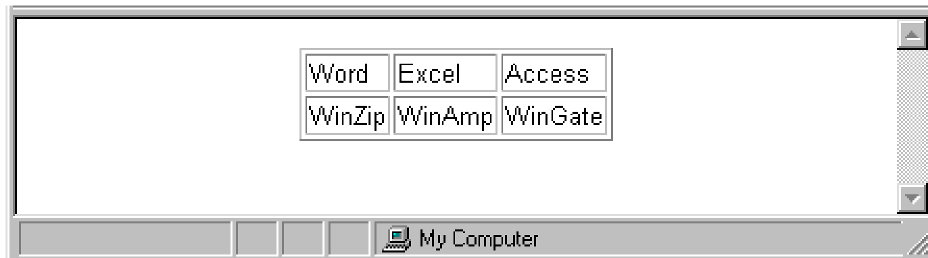
תרשים 7.2

אתה יכול לשנות את הערכים של המאפיין border כרצונך, כדי לקבל טבלאות עם גבולות (מסגרת) בעובי שונה.

יישור טבלה, התגית <table> והמאפיין align

קובץ **Table03.html** מראה את היישום של המאפיין **align** בתגית **<table>**. מאפיין align מיישר את הטבלה יחסית לגודל החלון בו היא מוצגת. כלומר, "align="left יצמיד את הטבלה לשמאל החלון (זוהי ברירת המחדל גם אם לא תציין את המאפיין), "align="center ימרכז את הטבלה המוצגת בחלון ו-"align="right יצמיד את הטבלה לצד ימין.

```
<table border="1" align="center">
```



תרשים 7.3

רוחב טבלה, התגית <table> והמאפיין width

רוחב הטבלה נקבע באופן אוטומטי לפי הערכים הנמצאים בתאים. יהיה יותר טקסט - הטבלה תהיה יותר רחבה. קובץ **Table04.html** מדגים כיצד רוחב כל עמודה נקבע לפי התא הארוך ביותר.



תרשים 7.4

כשאנו רוצים לשלוט על רוחב הטבלה, אנחנו יכולים להשתמש במאפיין **width**. הערכים של מאפיין זה יכולים להיות מספר פיקסלים או % מרוחב החלון בו מוצגת הטבלה. קובץ **Table05.html** מראה כיצד נקבע רוחב טבלה ל-80% מרוחב החלון:

```
<table border="1" align="center" width="80%">
```



תרשים 7.5

ומה קורה אם רוחב הטבלה שנקבע קטן מלהכיל את ערכי התא ברצף? שנה את אחוזי הרוחב מ-80% ל-20% וראה מה קורה.

מה יקרה אם רוחב הטבלה כולה יהיה 80% ותקטין את גודל החלון?

מילים לא יישברו (למשל, המילה Encyclopedia) אלא ייראו ברצף. מילה שאין לה מקום תעבור לשורה הבאה בתא, וגובה התא יסתדר בהתאם.

לחילופין (במקום לציין את רוחב הטבלה באחוזים), תוכל לציין את רוחב הטבלה בנקודות, למשל:

```
<table border="1" align="center" width="400">
```

כלומר, רוחב הטבלה יהיה 400 נקודות.

ומה קורה אם רוחב הטבלה שנקבע קטן מלהכיל את ערכי התא ברצף? שנה את רוחב הטבלה כולה מ-400 נקודות ל-200 נקודות וראה מה קורה.

מה יקרה אם רוחב הטבלה כולה יהיה 400 נקודות ותקטין את גודל החלון?

זכור שמרבית המשתמשים משתמשים ברזולוציה של 600x800 נקודות, כך שרוחב המסך העומד לרשותם הוא... 800 נקודות (כולל פס גלילה).

צבע רקע לטבלה, התגית <table> והמאפיין bgcolor

וקצת צבע?!?

קובץ **Table06.html** מראה טבלה עם צבע רקע. בבחירת צבע רקע יש לשים לב שתוכן הטבלה ייראה והצבע לא "יבלע" אותו.

```
<table border="1" align="center" width="80%" bgcolor="#ffe4b5">
```

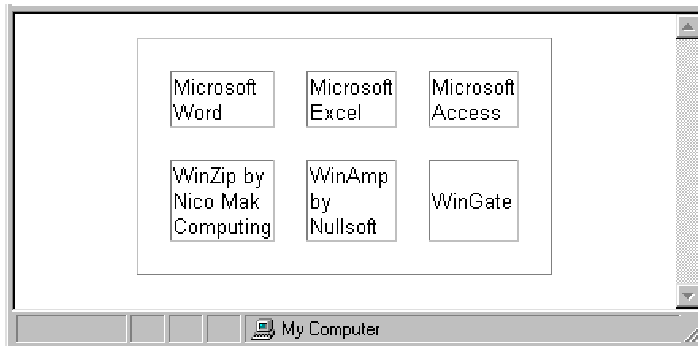


תרשים 7.6

מרווח בין תאים בטבלה, התגית <table> והמאפיין cellspacing

קובץ **Table07.html** מראה טבלה עם מרווח בין התאים הנקבע בנקודות.

```
<table border="1" align="center" width="50%" cellspacing="20">
```



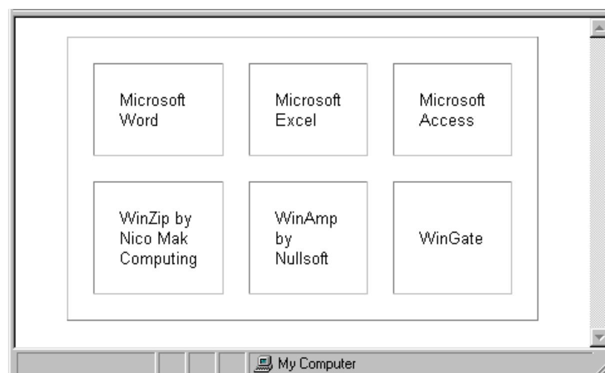
תרשים 7.7

ערך המאפיין **cellspacing** יכול להיות גם מוגדר באחוזים. אתה מוזמן לנסות ולראות מה יקרה. דע, שכאשר הרוחב נקבע באחוזים, השליטה עוברת לדפדפן הקובע את המרווח.

מרווח בין תוכן לגבול התאים בטבלה, התגית <table> והמאפיין cellpadding

קובץ **Table08.html** מראה טבלה בה נקבע מרווח בין תוכן הטבלה (במקרה זה מוצג טקסט אבל בהחלט יכולה להיות שם גם תמונה) לבין גבולות התא.

```
<table border="1" align="center" width="50%" cellspacing="20" cellpadding="20">
```



תרשים 7.8

ערך המאפיין **cellpadding** יכול להיות גם מוגדר באחוזים (%). אתה מוזמן לנסות ולראות מה יקרה. דע, שכאשר הרוחב נקבע באחוזים, השליטה עוברת לדפדפן הקובע את המרווח.

כעת אתה מוכן להתקדם הלאה ולהגדיר תאים!

תא בטבלה, התגית <td>

לטבלה מרכיב עיקרי הנקרא תא. התגית המגדירה תא נתונים היא <td>. הקבצים **Table01.html** ועד **Table08.html** הכילו תאים עם תוכן, כך שהנושא מוכר ולא חדש.

בטבלה להלן מוצגים מאפייני התגית <td>.

מאפיין	משמעות	ערכים
align	יישור תוכן בתא	**left**, center, right, justify
valign	יישור אנכי של התוכן בתא	top, **middle**, bottom
background	תמונת רקע לתא	gif
bgcolor	צבע רקע לתא	blue, green,.. #rrbbgg
colspan	מספר התאים בעמודה המשויכים לתא	number
rowspan	מספר התאים בשורה המשויכים לתא	number
height	גובה התא	pixels, %
width	רוחב התא	pixels, %
nowrap	ביטול גלישת טקסט בתא	אין ערכים

יישור תאים, התגית <td> והמאפיין align

המאפיין **align** מציין את סוג היישור שיוחל על הנתונים בתא. הוא יכול לקבל אחד מהערכים הבאים: center, justify, right או left.

* ערך ברירת המחדל של המאפיין **align** הוא left.
* הערך justify של המאפיין **align** בתגית **<td>** מציין כי הטקסט יהיה מיושר לשני צידי התא. כמובן שאם בתא שרוחבו גדול תהיה מילה אחת, לא יתבצע יישור דו-כיווני. סגנון זה מזכיר את סגנון היישור של ספר זה ושל טורי עיתון.

לדוגמה, קובץ **Table09.html** מראה את התאים בשורה הראשונה מיושרים ל**מרכז** (center) והתאים בשורה השנייה מיושרים ל**ימין** (right). שים לב שכל הטבלה מיושרת על ידי המאפיין align המוגדר בתגית <table>:



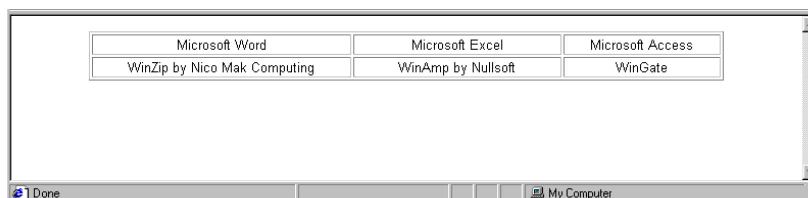
Microsoft Word	Microsoft Excel	Microsoft Access
WinZip by Nico Mak Computing	WinAmp by Nullsoft	WinGate

תרשים 7.9

רוחב התא, התגית <td> והמאפיין width

רוחב העמודה (רוחב כל התאים באותה עמודה) נקבע על פי התא הרחב ביותר בעמודה. קובץ **Table10.html** משרטט טבלה בה רוחב התא נקבע על ידי ציון מספר פיקסלים. כאשר משתמשים במאפיין **width** צריך לציין רק רוחב של תא אחד בעמודה:

```
<tr>
      <td align="center" width="250">Microsoft Word</td>
      <td align="center" width="200">Microsoft Excel</td>
      <td align="center" width="150">Microsoft Access</td>
</tr>
```



Microsoft Word	Microsoft Excel	Microsoft Access
WinZip by Nico Mak Computing	WinAmp by Nullsoft	WinGate

תרשים 7.10

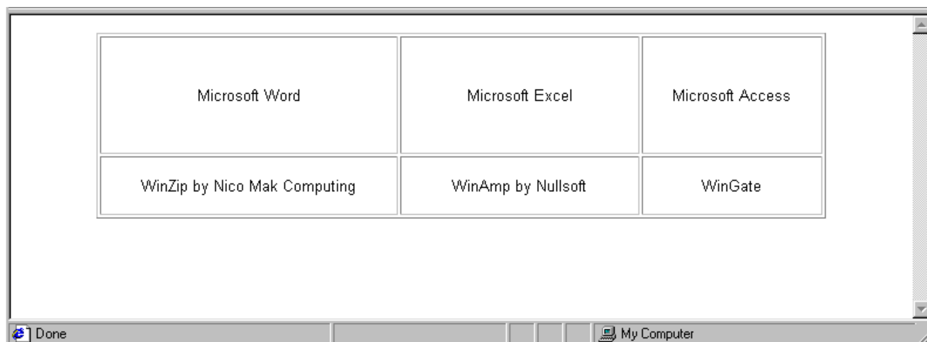
באותו אופן אפשר להגדיר את רוחב התא ב-% מגודל החלון (המסך).

גובה התא, התגית <td> והמאפיין height

גובה השורה נקבע באופן אוטומטי על פי התא הגבוה בשורה. קובץ **Table11.html** משרטט טבלה בה גובה התא נקבע על ידי ציון מספר פיקסלים. כאשר משתמשים במאפיין height צריך לציין רק גובה של תא אחד בשורה:

```
<html>
 <head>
  <title>Table11
  </title>
 </head>
 <body>
  <table border="1" align="center">
   <tr>
    <td align="center" width="250" height="100">Microsoft Word</td>
    <td align="center" width="200">Microsoft Excel</td>
    <td align="center" width="150">Microsoft Access</td>
   </tr>
   <tr>
    <td align="center" height="50">WinZip by Nico Mak Computing</td>
    <td align="center">WinAmp by Nullsoft</td>
    <td align="center">WinGate</td>
   </tr>
  </table>
 </body>
</html>
```

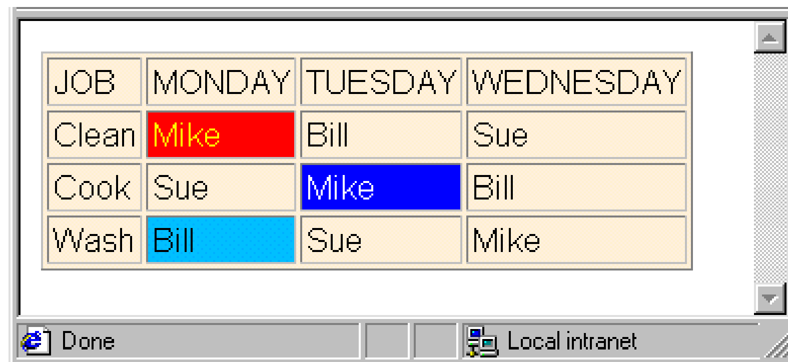
באותו אופן אפשר להגדיר את גובה התא ב-% מגודל החלון (המסך).



תרשים 7.11

צבע תא, התגית <td> והמאפיין bgcolor

מאפיין **bgcolor** מוכר מתגיות אחרות כמו <body> וגם <table>. השימוש בו במסגרת התגית **<td>** קובע את צבע הרקע לתא בודד אחד, לצורך הבלטה, לדוגמה. הקובץ **Table-Color1.html** מציג שימוש במאפיין זה.



תרשים 7.12

צבע הטבלה כולה נקבע על ידי התגית <table>, באופן הבא:

```
<table border="1" cellpadding="2" cellspacing="2" bgcolor="#fff8dc">
```

צבע תא נקבע על ידי התגית **<td>**, באופן הבא:

```
<td bgcolor="red">
```

שים לב ל-overide שעושות ההגדרות אחת על השנייה.

שורה בטבלה, התגית <tr>

שורה בטבלה מוגדרת על ידי התגית **<tr>**. גם לתגית <tr> יש מאפיינים הדומים מאוד לתגית <td> והם:

מאפיין	משמעות	ערכים
align	יישור אפקי של תוכן בתא	**left**, center, right, justify
valign	יישור אנכי של תוכן בתא	top, **middle**, bottom
bgcolor	צבע רקע לתא	blue, green,.. #rrbbgg

שים לב שמאפיינים שהוגדרו בתגית <tr> מתייחסים לשורה ולתאים. למשל, אם הגדרת את המאפיין align="center" תחת התגית <tr>, אין צורך להגדיר מחדש align="center" במסגרת התגית <td> באותה השורה, אלא אם אתה רוצה לשנות את היישור שנקבע ברמת השורה כולה (למשל, לתא מסוים אתה רוצה להגדיר align="left").

קובץ **Table12.html** דומה מאוד לקובץ **Table11.html**, פרט לכך שהגדרת היישור (מכיון שהיא זהה לכל התאים בשורה) מוגדרת ברמת השורה תחת התגית <tr> (בקובץ **Table12.html**) ולא ברמת התא תחת התגית <td> (קובץ **Table11.html**).

גם צבע ניתן להגדיר ברמת השורה, וקובץ **Table-Color2.html** מדגים זאת:

```
<tr bgcolor="yellow">
<tr bgcolor="#ADFF2F">
```

JOB	MONDAY	TUESDAY	WEDNESDAY
Clean	Mike	Bill	Sue
Cook	Sue	Mike	Bill
Wash	Bill	Sue	Mike

Done Local intranet

תרשים 7.13

טבלאות מורכבות

הטבלאות שהיכרנו עד כה היו פשוטות. פשוטות במובן, שהן היו טבלאות שבכל שורה היה אותו מספר של תאים.

מיזוג תאים בשורה, התגית <td> והמאפיין colspan

המאפיין **colspan** בתגית **<td>** קובע את מספר הטורים עליהם יחלוש תא מסוים. ערך **ברירת המחדל** הוא 1.

טיפ

ערך ברירת מחדל הוא ערך שמקבל מאפיין מסוים כאשר אינך מציין אותו במסגרת התגית. זה שהמאפיין לא נרשם במסגרת התגית זה לא אומר שהוא לא מופעל. הוא כן מופעל עם ערך ברירת המחדל שלו. במקרים רבים קיימים לדפדפן ערכים קבועים מראש בהם הוא משתמש להצגת מרכיבים שונים, אם אינו נתקל בהגדרה עבורם במסמך.

קובץ **Table13.html** מדגים שימוש במאפיין **colspan**:

```
<tr>
      <td></td>
      <td colspan="2">Platform Support</td>
</tr>
```

	Platform Support	
Manufacturer	**Mac**	**Windows**
GoLive	☺	☹
MacroMedia	☺	☺

My Computer

תרשים 7.14

בשורה הראשונה, התא הראשון הוא תא ריק: <td></td>.

בשורה הראשונה, נעשה מיזוג בין שני תאים:

```
<td colspan="2">Platform Support</td>
```

המשמעות היא, שבשורה בה נמצאת התגית <td> עם המאפיין colspan="2", יצורפו התא הנוכחי והתא שאחריו לתא **אחד** - זוהי משמעות המיזוג בין התאים באותה שורה.

השימוש במאפיין **colspan** לאו דווקא חייב לבוא בשורה הראשונה בטבלה. קובץ **Table14.html** מדגים טבלה מורכבת קצת יותר:

	Platform Support	
Manufacturer	**Mac**	**Windows**
GoLive	☺	☹
MacroMedia	☺	☺
Under $199		
SoftQuad	☹	☺

My Computer

תרשים 7.15

בשורה החמישית, נעשה מיזוג של כל שלושת התאים בשורה על ידי המאפיין "colspan="3. בנוסף, הטקסט בשורה זו עוצב בגופן (font) קטן ("size="2) וצמוד לתחתית התא ("valign="bottom), באופן הבא:

```
<tr>
	<td height="40" valign="bottom" colspan="3">
	  <font size="2">Under $199</font>
	</td>
</tr>
```

מיזוג תאים בעמודה, התגית <td> והמאפיין rowspan

המאפיין **rowspan** בתגית **<td>** קובע את מספר השורות עליהם יחלוש תא מסוים. ערך **ברירת המחדל** הוא 1.

קובץ **Table15.html** מראה טבלה כזו:

```
<html>
  <head>
    <title>Table15
    </title>
  </head>
  <body>
    <table border>
      <tr>
        <td>Course</td>
        <td>Subjects</td>
      </tr>
      <tr>
        <td rowspan="3">Course 825</td>
        <td>Windows 98</td>
      </tr>
      <tr>
        <td>Office 2000</td>
      </tr>
      <tr>
        <td>Internet Explorer 5.x</td>
      </tr>
      <tr>
        <td rowspan="2">Course 600</td>
        <td>C++</td>
      </tr>
      <tr>
        <td>Visual C++ 6</td>
      </tr>
```

```
    <tr>
      <td rowspan="4">WebMaster</td>
      <td>HTML</td>
    </tr>
    <tr>
      <td>JavaScript</td>
    </tr>
    <tr>
      <td>DHTML</td>
    </tr>
    <tr>
      <td>ASP</td>
    </tr>
  </table>
 </body>
</html>
```

Course	Subjects
Course 825	Windows 98
	Office 2000
	Internet Explorer 5.x
Course 600	C++
	Visual C++ 6
WebMaster	HTML
	JavaScript
	DHTML
	ASP

Done My Computer

תרשים 7.16

הסבר:

לפניך טבלה עם שתי עמודות ו-10 שורות. נכון שבעמודה השמאלית יש רק ארבע שורות אבל בעמודה הימנית יש 10 שורות. מעתה ואילך נספור גם את השורות משמאל וגם את השורות מימין, ויהיה עליך לעקוב בדייקנות אחר ההסבר.

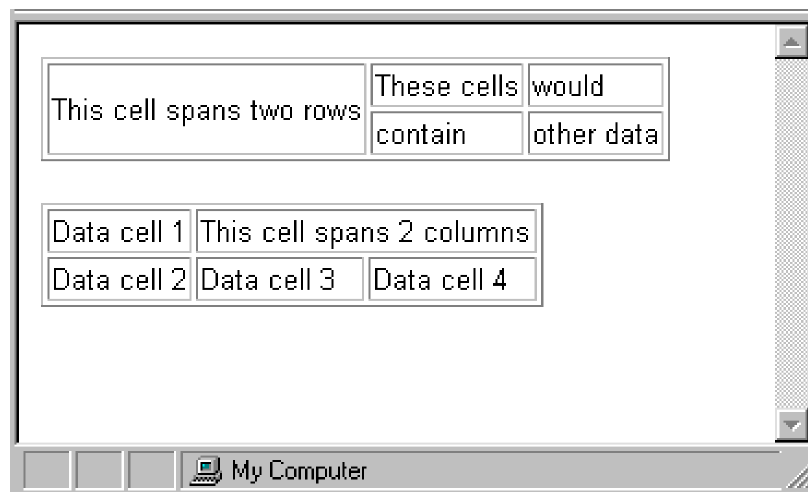
בשורה הראשונה (לפי עמודה שמאלית) מוגדרים שני תאים תוך שימוש בתגית <td>.

בשורה השנייה (לפי עמודה שמאלית) מוגדרים גם כן שני תאים, אבל התא הראשון מוגדר עם המאפיין "rowspan="3. כלומר, התא יתפוס שלוש שורות. שים לב שאחרי הגדרת שורה זו יש הגדרה של עוד שתי שורות נוספות (ימין), אבל הפעם עם תא אחד בלבד בכל שורה. היכן התא השני? ובכן, תא אחד כבר הוגדר על ידי המאפיין **rowspan** ולכן נותר להגדיר בכל שורה רק תא אחד (מתוך שניים).

בשורה השלישית (לפי עמודה שמאלית) מוגדרים שני תאים, אבל התא הראשון מוגדר עם המאפיין "rowspan="2, שמשמעותו, שתא זה יתמזג עם התא שמתחתיו. עכשיו צריך להגדיר שורה נוספת אבל עם תא אחד בלבד (מתוך שניים).

בשורה הרביעית (לפי עמודה שמאלית) מוגדרים גם כן שני תאים, אבל התא הראשון מוגדר עם המאפיין "rowspan="4. כלומר, התא יתפוס ארבע שורות. שים לב, שאחרי הגדרת שורה זו יש הגדרה של עוד שלוש שורות נוספות (ימין) אבל הפעם עם תא אחד בלבד בכל שורה. היכן התא השני? ובכן, תא אחד כבר הוגדר על ידי המאפיין **rowspan** ולכן נותר להגדיר בכל שורה רק תא אחד (מתוך שניים).

בקובץ **TableSpan.html** תמצא שתי דוגמאות נוספות לבניית טבלאות בעזרת **rowspan** ו-**colspan**:



תרשים 7.17

פרק 8

טופס

בעוד שקישורים מסייעים למשתמש באתר להשתמש בו ביתר קלות, טפסים מאפשרים לו להשתתף בחווית האתר.

זו דרך מצוינת לקבל משוב מהמבקרים.

טפסים ברשת דומים מאוד לטפסי נייר - הם בנויים מרווחים המיועדים לכתיבה, תיבות סימון ורשימות שמהן צריך לבחור. ההבדל הוא, שטופס נייר צריך להישלח בדואר, בפקס או להימסר לידיו של מישהו, בזמן שטפסים ברשת נשלחים בדואר אלקטרוני ומגיעים במהירות ליעדם. במונחים טכניים יותר, טפסים מגדילים את כוחו של מפתח האתר והופכים את האתר לאינטראקטיבי.

טפסים עוזרים לאתר שלך להיראות מקצועי ולהיות אינטראקטיבי. הטפסים מספקים מספר דרכים לקבל ולארגן את המידע מהמשתמש. להלן מספר דוגמאות לשימוש בטפסים: הרשמה למועדון לקוחות, שאלון, זיהוי משתמש (סיסמה), שאילתה לבסיס נתונים ועוד. הוספת טפסים לאתר שלך הופכת אותו למשהו שונה ממצגת של צד אחד.

נסה את טופס ההרשמה למועדון לקוחות הוד-עמי בכתובת:

http://www.hod-ami.co.il/store/guest.htm

תרשים 8.1

מרכיבים בסיסיים של טופס

לטפסים יש ארבע תגיות ונעשה בהן שימוש כמו בכל תגית אחרת ב-HTML. התגית הראשונה, **<form>**, היא תגית תחום המגדירה את תחילתו ואת סופו של הטופס, ובנוסף היא מגדירה כיצד יישלחו הנתונים שנכתבים בטופס ולהיכן. שלוש התגיות האחרות מהוות את החלק בטופס בו מקליד המשתמש נתונים: **<textarea>** שדה טקסט, **<select>** רשימות הבחירה ו-**<input />** שדות טקסט, תיבות סימון, אפשרויות ולחצנים עליהם לוחץ המשתמש.

> **טיפ**
>
> רוב תוכנות העריכה הקיימות ברשת תומכות ביצירת טפסים, ותוכל להשתמש בהן אם תרצה. רוב החומר שתלמד בהמשך הפרק יכול להיעשות באופן אוטומטי על ידי רוב התוכנות ליצירת טפסים, אך חשוב מאוד לדעת כיצד עובדים הדברים, ורק אחר כך לשחק עם הכלים שחוסכים זמן.

יצירת טופס, התגית <form>

התגית **<form>** משמשת לציון תחילת הטופס, והתגית המסיימת **</form>** משמשת לציון המקום בו מסתיים הטופס. תגיות הטופס האחרות מעובדות על ידי הדפדפן, רק בתחום התגית <form>. לכן, עליך להגדיר בדיוק רב היכן הטופס שלך מתחיל, והיכן הוא נגמר.

> **טיפ**
>
> הוסף את התגית **</form>** מייד לאחר התגית **<form>** ורק אחר כך חזור למלא את שאר הטופס. כך, תחסוך מעצמך כאבי ראש מיותרים ותוכל להיות בטוח שלא שכחת להציב תגית סיום.

מאפיינים לתגית **<form>**:

מאפיין	משמעות	ערכים
action	לאן יישלח המידע שבטופס	URL
method	כיצד יישלח המידע שבטופס	get, post

לתגית **<form>** יש שני מאפיינים עיקריים המגדירים כיצד יתנהג הטופס. שני מאפיינים אלה קובעים לאן יישלח המידע שבטופס, וכיצד הוא יישלח לשם.

המאפיין הראשון הוא **action**. מאפיין זה מגדיר את כתובת ה-URL אליה יישלח המידע מהטופס. הוא מופיע בתגית <form>, כך:

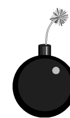
```
<form action="url">
הטופס עצמו
</form>
```

כתובת URL יכולה להיות כל כתובת אתר, אך אם תרצה שהמידע בטופס יעובד כהלכה, היא צריכה להיות כתובת של CGI Script או ASP Script (קובץ עם סיומת asp) שתוכנן לטפל במידע הנשלח מטופס זה. תוכל להתייחס לקובץ ASP כאל תוכנית היושבת בשרת ומחכה לנתוני הטופס. אותה תוכנית יודעת לעבד את הנתונים ובהתאם להחזיר תשובה. כתובת URL יכולה להיות גם כתובת דואר אלקטרוני אליה יישלחו הפרטים שמילא המשתמש. החלק הרביעי והאחרון של ספר זה עוסק ב-ASP.

המאפיין השני של תגית **<form>** הוא **method**. מאפיין זה מגדיר כיצד יישלח המידע שבטופס לכתובת המצוינת במאפיין action. המאפיין יכול לקבל אחד משני ערכים: get או post. השיטה get היא שיטה פשוטה לקבלת נתונים לעומת post המאפשרת משלוח מידע רב יותר. הדרך בה כתוב קובץ ASP שתפקידו לעבד את הנתונים, תקבע באיזו שיטה עליך להשתמש. למאפיין method אין השפעה על הטופס עצמו, רק על הדרך בה נשלחים הנתונים לקובץ ASP.

השימוש במאפיין **method** בתגית **<form>** נעשה כך:

```
<form action="newvisitor.asp" method="post">
הטופס עצמו
</form>
```



אזהרה

למרות שאפשר לוותר על המאפיין method והוא יעבוד כראוי, זהו לא רעיון טוב בכלל. רצוי שתהיה מדויק ככל האפשר כשאתה מתעסק ב-HTML, גם כדי שתזכור מה רצית לעשות במקרה מסוים, וגם כי הגדרות ברירת המחדל עלולות להשתנות בעתיד או להיות שונות בדפדפנים שונים.

אז באיזו מהשיטות להשתמש למשלוח הטפסים שלך? לשיטה **get** יש כמה מגרעות והן: המשתמש רואה את הפרמטרים שנשלחו בשורת הכתובת (לא תמיד זה רצוי). זו אינה שיטה בטוחה להעברת מידע ולבסוף העברת הטקסט מוגבלת ל-1,000 תווים בערך. אבל, היתרון שלה הוא בעיקר עם מנועי חיפוש והוספת החיפוש (השאילתה) ל**מועדפים** (Favorites). בשיטת **post** אין חשיפת מידע בשורת הכתובת ואין מגבלה על מספר התווים. לסיכום, שיטת **post** עדיפה לשימוש.

טיפ

נתוני הטופס לא חייבים להישלח, ולכן תוכל שלא לציין את המאפיין **action**. בחלק הרביעי של ספר זה בנושא ASP תלמד כיצד לבנות תוכנית ASP שתוכל לקלוט את נתוני הטופס. עד אז, אם ברצונך לשלוח את נתוני הטופס, תוכל לציין במאפיין **action** את כתובת הדואר האלקטרוני שלך (רשום במקום moki@smoopi.co.il את הכתובת האלקטרונית שלך):

```
<form method="post" action="maito:moki@smoopi.co.il">
```

אזהרה

לצערי, W3C לא הגדיר במדויק את המאפיין **action** העושה שימוש בערך :mailto. דבר הגורם לכך, שהדפדפן עשוי שלא להפעיל את תוכנת הדואר ו/או להפעיל את תוכנת הדואר, אבל לא להעביר אליה נתונים וכדומה.

הכי בטוח זה לשלוח את הטופס לקובץ ASP שבשרת, ועל כך בחלק הרביעי של ספר זה.

אזהרה

טופס שנשלח בעזרת mailto אינו מאובטח.

בדרך כלל, תגדיר את שני המאפיינים של תגית **<form>** בו-זמנית, כך:

```
<form method="post" action="mailto: your email address">
הטופס עצמו
</form>
```

טקסט חופשי, התגית <textarea>

התגית **<textarea>** מאפשרת למשתמש להקליד טקסט חופשי. אזורי טקסט מוגדרים בעזרת תגית פתיחה <textarea> ומסתיימים בתגית סגירה </textarea>. ניתן לציין טקסט ברירת מחדל שיופיע באזור הטקסט, ראה קובץ **Form-TextArea1.html**:

```
<textarea name="freetext1">
      Enter your remarks here!
</textarea>
```

דוגמת קוד זו יוצרת את תרשים 8.2.



תרשים 8.2: כטקסט ראשוני רשום מה שאתה רוצה, המשתמש יוכל למחוק את הכתוב אם ירצה.

כמו לתגית **<form>**, כך גם לתגית **<textarea>** יש מאפיינים מיוחדים שיופיעו בתגית עצמה:

מאפיין	משמעות	ערכים
name	שם	data
cols	מספר התווים בשורה (רוחב)	number
rows	מספר השורות (גובה)	number
disabled	לא ניתן לגשת לשדה	
readonly	לא ניתן לשנות את תוכן השדה	
tabindex	מספר סידורי המציין את סדר השדות שבטופס	number

המאפיין הראשון הוא **name**, והוא חובה. שם אזור הטקסט הוא השם שיוצמד לנתונים שיתקבלו מאזור זה בטופס, כשהמשתמש יישלח אותו. אתה חייב לתת שמות לכל חלקי הטופס, מכיון שרק כך יידע קובץ ASP כיצד לפענח את המידע.

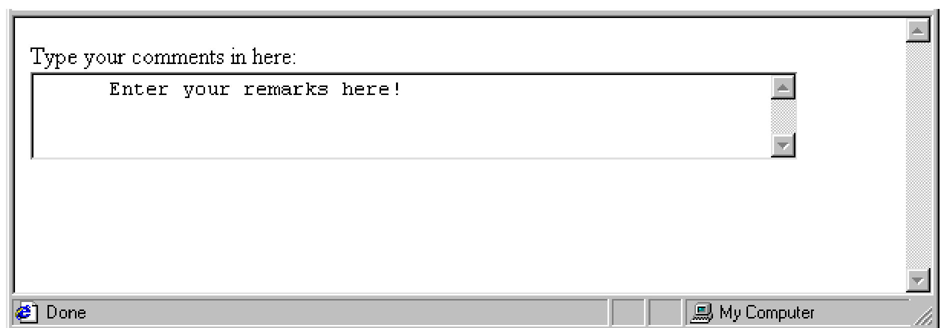
הערה

לכל שדה בטופס ניתן לתת ערך למאפיין **tabindex** המציין את סדר השדות שבטופס. בדרך כלל, ממלאים את הטופס מלמעלה למטה, ומשמאל לימין. כך זה בטופס בשפה האנגלית ואז אין צורך לציין מאפיין זה. מכיון שטופס בעברית ממלאים מימין לשמאל יש לציין מאפיין זה ועל כך בפרק 9.

שני מאפיינים נוספים הם **rows** ו-**cols**, כל אחד מהם מגדיר את גודל אזור הטקסט באותיות, בשורות ובעמודות. אם לא תציין מאפיינים אלה, יקבל המאפיין rows את ערך ברירת המחדל 1, והמאפיין cols יקבל את ערך ברירת המחדל 20, דבר שיאפשר אזור צפייה מוגבל. אפשר לשנות זאת, כפי שכבר צוין. ראה קובץ **Form-TextArea2.html**:

```
Type your comments in here:<br>
    <textarea name="comments" rows="4" cols="60">
      Enter your remarks here!
    </textarea>
```

תוכל לראות את תוצאות קוד זה בתרשים 8.3:



תרשים 8.3: אזור הטקסט נקרא comments (לפי ערך המאפיין name) והוא ברוחב 60 תווים ובגובה 4 שורות.

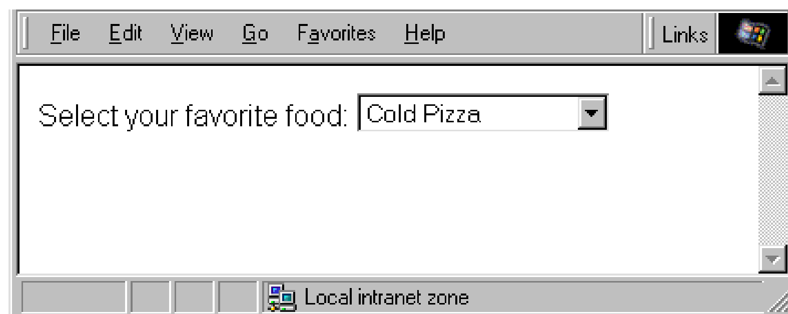
רשימה נפתחת, התגית <select>

למרות שתיבות טקסט חופשי מאפשרות למשתמשים להתבטא באופן חופשי, לעיתים תרצה שהמשתמש יבחר באחת מאפשרויות התגובה שאתה מספק לו. במקרה כזה עליך להשתמש במרכיב שונה של טופס - תיבת רשימה נפתחת. תיבת רשימה נפתחת מיוצגת על ידי התגית **<select>**.

מאפיין	משמעות	ערכים
name	שם	data
size	הגדרת גובה הרשימה	number
tabindex	מספר סידורי המציין את סדר השדות בטופס	number

התגית עצמה פשוטה מאוד ומכילה מאפיינים מעטים. הלחצן עצמו יהיה מוגדר בין תגית פתיחה **<select>** לתגית סיום **</select>**. אלה המאפיינים:

- **name** (כמו ב-<textarea>) מגדיר את השם שיזהה את הנתונים, שאותם יקבל קובץ ASP מרכיב זה של הטופס.
- **size** מגדיר את גובה הרשימה במסך. אם תשאיר מאפיין זה ריק, או שתתן לו ערך 1, תמצא על המסך רק אפשרות אחת ותיבת הבחירה תיפתח ברגע שילחץ עליה המשתמש. ערך גבוה מ-1 יציג יותר אפשרויות על המסך. ראה קובץ **Form-Select1.html** ותרשים 8.4.



תרשים 8.4: כשתלחץ על החץ ליד הרשימה, יופיע תפריט ובו האפשרויות.

לאחר שהגדרת את התגית **<select>**, עליך להגדיר את האפשרויות שיש למשתמש. תגית **<option>** מגדירה כל אפשרות ספציפית שהמשתמש יראה. התגית מזוהה רק בין שתי התגיות <select> ו-</select>.

הנה דוגמה הלקוחה מקובץ **Form-Select1.html**:

```
<form method="get" action="mailto: your email address">
    Select your favorite food:
    <select name="food">
      <option>Cold Pizza</option>
      <option>Cold Chinese</option>
      <option>Cold Fried Chicken</option>
    </select>
</form>
```

לתגית **<option>** יש שני מאפיינים:

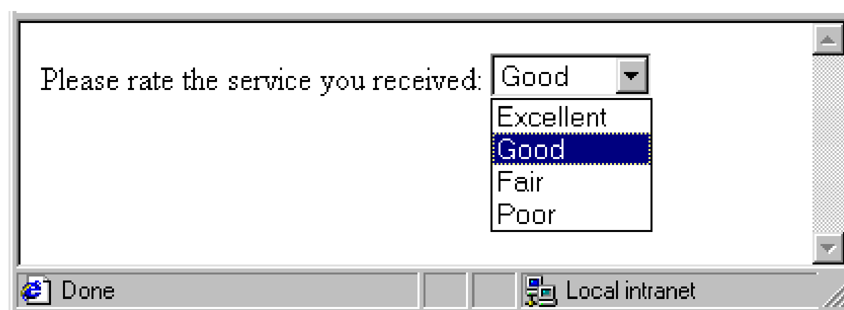
מאפיין	משמעות	ערכים
selected	הבחירה שתוצג לראשונה	
value	ערך הבחירה	data

- **value** הוא מה שנשלח עם שם האפשרות כשהמשתמש בוחר בה. ASP script הכתוב כחלק מדף ASP, משתמש בערך המוגדר על ידי value לעיבוד הנתונים ולא בטקסט המופיע ברשימה עצמה. שימוש חכם באפשרות זו יכול להעביר לדף ASP נתונים פשוטים לשימוש. אם לא תשתמש במאפיין זה, ערך ברירת המחדל יהיה הטקסט שהצבת באפשרות עצמה, בדיוק כמו בדוגמה שלמעלה.
- **selected** הוא מאפיין ללא ערך. הוא מגדיר את האפשרות (או אפשרויות במקרה של <select multiple>) שתהיה מסומנת ברשימה, כברירת מחדל. לרוב, כדאי לסמן מראש את האפשרות השכיחה ביותר.

לדוגמה, אם תרצה שלקוחותיך ידרגו את רמת השירות של עובדיך, תשתמש ברשימה כזו, הלקוחה מקובץ **Form-Select2.html**:

```
<form action="mailto: your email address" method="get">
     Please rate the service you received:
     <select name="service">
       <option value="100">Excellent</option>
       <option value="75" selected>Good</option>
       <option value="60">Fair</option>
       <option value="50">Poor</option>
     </select>
</form>
```

שים לב, שערכי האפשרויות מתייחסים לשיטת ניקוד ולא לטקסט המופיע ברשימה בפועל. הרשימה עצמה תיראה בדפדפן כמו שמוצג בתרשים 8.5.



תרשים 8.5: אתרי Web יכולים לנתב מידע עבורך, כמו גם לספק מידע לאנשים המבקרים באתר.

קבלת מידע מהמשתמש, התגית <input />

התגית הבאה שלנו לטפסים מאפשרת קבלת סוגים שונים של מידע מהמשתמש. זה יכול להיות טקסט בדומה לתגית **<textarea>**, סיסמה או קלט בצורת שאלון.

מאפיינים:

מאפיין	משמעות	ערכים
name	שם לשדה	text
type	סוג השדה	text, password, checkbox, radio, submit, reset, file, hidden, image, button
alt	תיאור קצר עם הסמן	text
size	גודל השדה	number
value	ערך שיוצב בשדה	data
checked	ברירת המחדל למאפיינים radio ו-checkbox	
disabled	ביטול הגישה לשדה	
readonly	לצפייה בלבד	
maxlength	מספר תווי הקלט המקסימלי	number
tabindex	הגדרת סדר המעבר בין השדות	number
src	הגדרת התמונה המשויכת במידה ונבחר type=image	URL

המאפיינים היחידים שנדרשים הם **type** ו-**name**.

שדה טקסט, התגית <input /> והמאפיין "type="text

אחד הערכים האפשריים למאפיין **type** של התגית **<input />** הוא **text**, היוצר תיבת טקסט בת שורה אחת באורך שאתה קובע. שים לב, אורך התיבה והאורך המקסימלי יכולים להיקבע בנפרד על ידך. אפשר שתהיה תיבה ארוכה יותר (או קצרה יותר) מכמות התווים שמותר למשתמש להקליד לתוכה. דוגמה לתגית **<input />** המגדירה תיבת TEXT:

```
Last Name: <input type="text" name="last_name" size="40" maxlength="40" />
```

אם תרצה, תוכל להשתמש במאפיין **value** בתוך הגדרת תיבת הטקסט כדי לקבוע את ערך ברירת המחדל לתיבת הטקסט שהצבת בתוך הטופס, כמו בקובץ **Form-Text.html**:

```
<form method="get" action="mailto: your email address">
   Type of Proccessor:
   <input type="text" name="processor" size="50"
    maxlength="50" value="Pentium" />
</form>
```

בתרשים 8.6 תוכל לראות את התוצאה בדפדפן **Internet Explorer**:



תרשים 8.6: השימוש בערך text של המאפיין type בתגית <input />.

תיבת סימון, התגית <input /> והמאפיין "type="checkbox

הערך **"type="checkbox** מוסיף לטופס שלך תיבות סימון. אפשרות זו טובה למקרים בהם לשאלה מסוימת יכולות להיות מספר תשובות נכונות. תוכל לקבוע תיבה מסוימת שתהיה מסומנת מראש, על ידי הוספת המאפיין **checked** להגדרת תיבת הסימון המבוקשת. אם המשתמש לא מסכים עם התשובה, עליו לבטל את הסימון.

דוגמה לקוד תיבות סימון הלקוחה מקובץ **Form-CheckBox.html**:

```
<form action="mailto: your email address" method="get">
    Types of Opertaring System you own:
    <input type="checkbox" name="Win2000" value="win2k" />Windows 2000
    <input type="checkbox" name="Win98" value="win98se" checked />Windows 98
    <input type="checkbox" name="Win95" value="Windows95" />Windows 95
    <input type="checkbox" name="Linux" value="Linux" />Linux 6.0
</form>
```

באפשרות זו ניתן לבחור במספר אפשרויות בו-זמנית. מכיון שכך, כל תיבת סימון מוגדרת בשם שונה. שים לב כיצד זה נראה בדפדפן:

Types of Opertaring System you own: ☐ Windows 2000
☑ Windows 98 ☐ Windows 95 ☐ Linux 6.0

Done | Local intranet

תרשים 8.7: שים לב כי האפשרות Windows 98 נבחרה מראש.

אפשרויות, התגית <input /> והמאפיין "type="radio

כמו באפשרות checkbox, גם ב-**radio** אתה מאפשר למשתמש לבחור מבין מספר אפשרויות מוגדרות מראש. בניגוד ל-checkbox, לחצני האפשרויות שהגדרת יאפשרו קבלת תשובה אחת בלבד. מכיון שכך, בכל הלחצנים יוגדר אותו שם (המאפיין name). המרכיבים הבסיסיים של לחצני אפשרויות זהים לאלה של תיבות סימון.

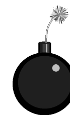
כמו ברשימה נפתחת, האפשרות radio דורשת שכל מרכיב יקבל ערך שיוגדר במאפיין value ויישלח לקובץ ASP לעיבוד, במקום הטקסט שמופיע בדפדפן. חשוב שערכי המרכיבים (המאפיין value) בשדה לחצני האפשרויות יהיו שונים. שים לב לדוגמה שלפניך, הלקוחה מקובץ **Form-Radio.html**:

```
<form method="post" action="mailto: your email address">
    Choose the software type you use most often:<br>
    <input type="radio" name="Software" value="W" checked /> Word
    <input type="radio" name="Software" value="E" /> Excel
    <input type="radio" name="Software" value="7" /> WinZip
    <input type="radio" name="Software" value="O" /> Other
</form>
```

Choose the softwaretype you use most often:
◉ Word ○ Excel ○ WinZip ○ Other

Done | Local intranet

תרשים 8.8



אזהרה

באפשרות radio כדאי לציין לחצן ברירת מחדל, מכיון שייתכן שמשתמש יידלג על תיבת לחצני האפשרויות מבלי לסמן אפשרות. תוכל לסמן מראש את אחת האפשרויות, על ידי הוספת הערך checked להגדרת הלחצן, וכך למנוע בעיות בעיבוד הטופס על ידי ASP.

ניקוי טופס, התגית <input /> והמאפיין type="reset"

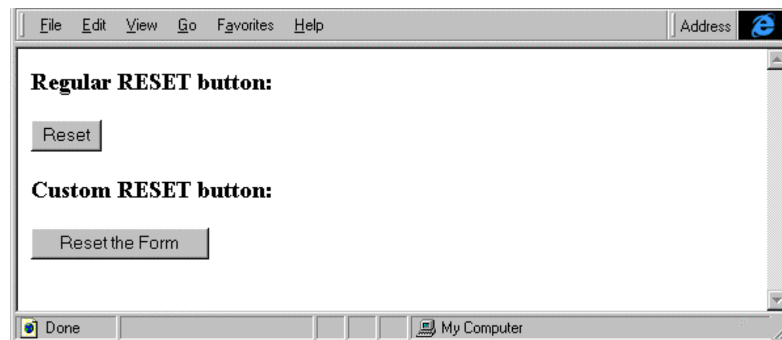
לתוך תגית **<input />** נבנתה היכולת "לנקות" טופס HTML. האפשרות **reset** יוצרת לחצן (שמו נקבע על ידי מחרוזת value), שמאתחל את כל המרכיבים בטופס המסוים לערכי ברירת המחדל (מוחק את כל המידע אותו הקליד המשתמש). דוגמה לכך:

```
<input type="reset" />
```

בתוספת להצהרת **value** תוכל להוסיף כותרת ללחצן, כך:

```
<input type="reset" value="Reset the Form" />
```

את התוצאה ניתן לראות בתרשים 8.9.



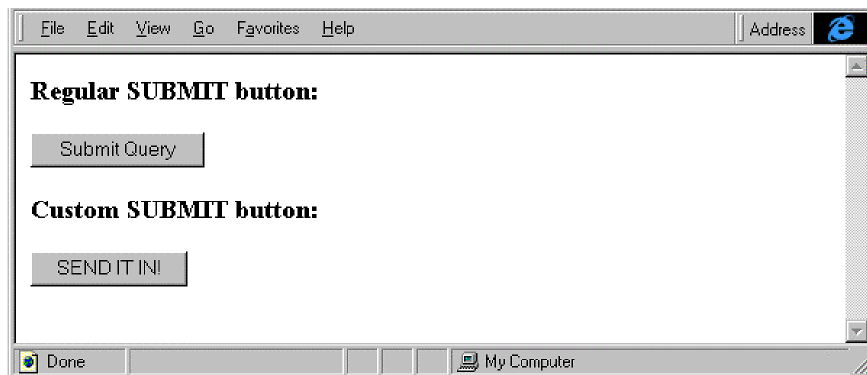
תרשים 8.9: לחצני RESET עם המאפיין value וללא המאפיין value.

שלח טופס, התגית <input /> והמאפיין type="submit"

לתגית **<input />** יש מאפיין ששולח באופן אוטומטי את הנתונים שהוקלדו לטופס HTML. המאפיין **submit** יכול לקבל ערך value, שמשמש לשינוי שם הלחצן. המטרה היחידה של לחצן **submit** היא לשלוח את כל המידע הנוסף שהוקלד על ידי המשתמש.

ראה את הדוגמאות הבאות ואת תרשים 8.10 :

```
<input type="submit" />
<input type="submit" value="SEND IT IN!" />
```



תרשים 8.10 : לחצני SUBMIT ללא/עם המאפיין value.

כמחרוזת עבור המאפיין **value** תוכל לכתוב כל דבר, אך כדאי לזכור שמילים "קטנות", למשל OK, לא נראות הכי טוב על לחצן. כדי לגרום ללחצן להיראות גדול יותר, הוסף למחרוזת value רווחים משני צידי המילה. ראה דוגמה.

```
<input type="submit" value="        go          " />
```

דוגמה

כעת נחבר את כל מרכיבי הטופס בדף אחד. לצורך העמדה של תיבות הטקסט, תיבות הסימון ותיבות האפשרויות ניעזר בטבלה ובקצת צבע.

הטופס בקובץ **Subscribe.html** מחולק למספר חלקים שבהם תוכל להבחין בקלות, תוך כדי התבוננות בקוד :

פתיחת הטופס, התגית **<form>** באמצעות המאפיין **action** מפנה את הטופס לדואר האלקטרוני של הוצאת הוד-עמי. כתובת ההוצאה (שדה To) היא info@hod-ami.co.il. בשדה Subject של ההודעה יופיע הטקסט Hod-ami Web Club. לצערי, W3C לא הגדיר במדויק את המאפיין action העושה שימוש בערך :mailto. דבר הגורם לכך, שהדפדפן עשוי שלא לשלוח דואר (כפי שנרצה), אלא הוא יפעיל את תוכנת הדואר אצל המשתמש עם ערכים בשדות To ו-Subject (מתגית <form>) אבל ללא ערכי הטופס - הודעה ריקה, כך בכל אופן זה קורה אם תוכנת הדואר היא Eudora, למשל. מי שמשתמש בתוכנת Outlook Express לא ייתקל בבעיה. שים לב, שטופס שנשלח בעזרת mailto אינו מאובטח.

```
<form name="myForm" method="post"
    action="mailto:info@hod-ami.co.il?SUBJECT=Hod-Ami Web Club">
```

פרק 9

HTML בעברית זה LMTH

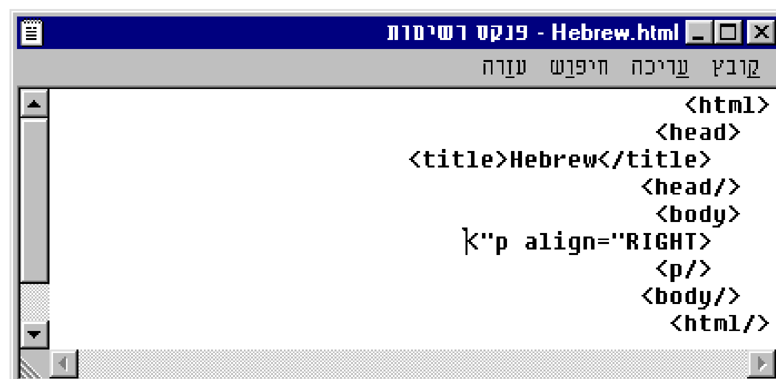
רוב העולם כותב משמאל לימין - אצלנו זה הפוך, והאלף-בית שלנו לא מוכר לכולם (כדי להקשות...). אתה, בתור מפתח אתרים באינטרנט, צריך להתמודד עם בעיות אלו.

מכיון שעברית היא השפה בה רובנו שולטים טוב ביותר, ומכיון שזו השפה הרשמית בארץ, אתרים רבים מפותחים בשפה זו. פיתוח דפי HTML בעברית אינה פעולה קלה במיוחד, אך גם אינו צריך להוות מכשול רציני בפני מפתח ברמה סבירה.

בעיקרון, קיימים שני תקנים עיקריים לשפה העברית באינטרנט: עברית **חזותית** (Visual) ועברית **לוגית** (Logical). Netscape תומכת בעברית חזותית בעזרת גופנים מיוחדים שנוצרו עבורה (Globes או Web AD Hebrew הם המוכרים ביותר), ודפדפני האינטרנט של Microsoft תומכים בעברית לוגית **באופן מובנה** (Built In), ללא צורך בהתקנת גופנים או התקנה אחרת.

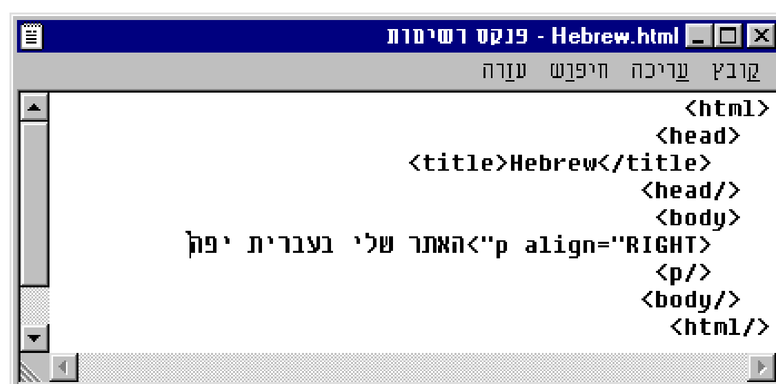
```
<html>
  <head>
    <title>Hebrew</title>
  </head>
  <body>
    <p align="right">
    </p>
  </body>
</html>
```

הצג את הסמן בסוף השורה השישית ולחץ Ctrl + Right Shift, כך שהטקסט ייראה מימין לשמאל באופן הבא:



תרשים 9.1

רשום לדוגמה ״האתר שלי בעברית יפה״.



תרשים 9.2

שמור את הקובץ והצג אותו בדפדפן Internet Explorer.

אם השורה בעברית נראית כך: **הפי תירבעב ילש רתאה**, יהיה עליך להציב את הסמן בשטח חלון התוכן של הדפדפן, ללחוץ לחיצה ימנית בעכבר, מתוך תפריט הקיצור לבחור **קידוד** (encoding) ולבחור באחד מתקני העברית: **Hebrew (ISO-Logical)** או **Hebrew (Windows)** כדי לקרוא את העברית כנדרש.

כדי שהדברים ייעשו באופן אוטומטי על ידי דפדפן IE, יהיה עליך להוסיף את המשפט הבא בין התגיות **<head>** ל-**</head>**:

```
<meta http-equiv="content-type" content="text/html;charset=windows-1255" />
```

הגדרה זו נקראת **Hebrew(Windows)**,

או

```
<meta http-equiv="content-type" content="text/html;charset=iso-8859-8-i" />
```

הגדרה זו נקראת **Hebrew(ISO-Logical)**.

משפט זה מורה לדפדפן Internet Explorer שהדף כתוב בעברית לוגית, והוא יציג אותו כפי שצריך באופן אוטומטי ללא התערבותנו כמשתמשים (ראה קובץ **HebrewIE.html**).

המאפיין **dir** מגדיר את כיוון הכתיבה. אנחנו כמובן צריכים לכתוב **"dir="rtl**. את מאפיין זה ניתן לשלב בתגיות: <font>, <p>, <td> וכדאי להתחיל מהתגית <html> באופן הבא: <html dir="rtl">.

טפסים בעברית

חשוב לזכור שחלק חשוב באינטרנט הוא נושא העברת המידע. כמעט כל אתר המכבד את עצמו מבקש מהגולש אליו את פרטיו לצורך זה או אחר (למשל, רישום למועדון הלקוחות של הוצאת הוד-עמי), אבל כדי למלא טפסים בעברית כדאי לנהוג בסבלנות. הפעלה של קובץ **HebrewForm.html** תציג טופס. אם תנסה לדלג בין שדות הטופס בעזרת מקש Tab, תראה איך לאחר שדה **כתובת**, הסמן מדלג לשדה **ישוב** ולחיצה נוספת מביאה את הסמן לשדה **מיקוד**. לעיתים, דילוג באמצעות מקש Tab (דרך חוקית לגמרי) בין שדות טופס עשוי להעביר אותך לשדה בקצה השני של הטופס, דבר שלא התכוונת אליו. מאחר שברירת המחדל של הדפדפן היא **משמאל לימין**, יש לעצב את הטופס ולבנות אותו בהתאם.

המאפיין tabindex

כתוב את השדות לפי הסדר שתרצה שהם יופיעו על המסך. בתוך תגית **<input />** של הטופס הוסף את המאפיין **tabindex**, והגדר את מספור דילוג ה-Tab עבור כל השדות בטופס. כעת, קוד המקור של הטופס ייראה כך (שים לב למאפיינים tabindex שבגוף הקוד, ובעיקר לכך שהם אינם בסדר רציף). ראה בקובץ **HebrewForm.html** (שים לב שקוד המקור של הקובץ מופיע בספר באופן מקוטע רק לצורך המחשה):

```
<input name="zip" type="TEXT" style="text-align: right" size="5"
        align="right" tabindex="4" />
```

אם תרצה ליצור טופס, בו המשתמש כותב בעברית, כדאי שתשים לב גם למיקום הסמן כאשר הוא נכנס לשדה הטופס. כדי להציב את הסמן בקצה הימני של שדה הטופס (בכתיבת טקסט, כגון שם או כתובת) עליך להשתמש במאפיין נוסף של התגית **<input />**.

```
<input name="cert" type="text" style="text-align: right" size="10" align="right" />
```

באופן זה אתה שולט בצד בו יוצב הסמן, בעת הכניסה לשדה הטופס. הדבר בו אינך שולט הוא - השפה. לשם כך, נועדה ההערה המקדימה שהצבנו באתר - **לחץ Alt+Right Shift כדי לכתוב בעברית** (הקשה על צירוף המקשים Alt+Right Shift משנה את השפה לעברית, ואילו Alt+Left Shift משנה את השפה לאנגלית).

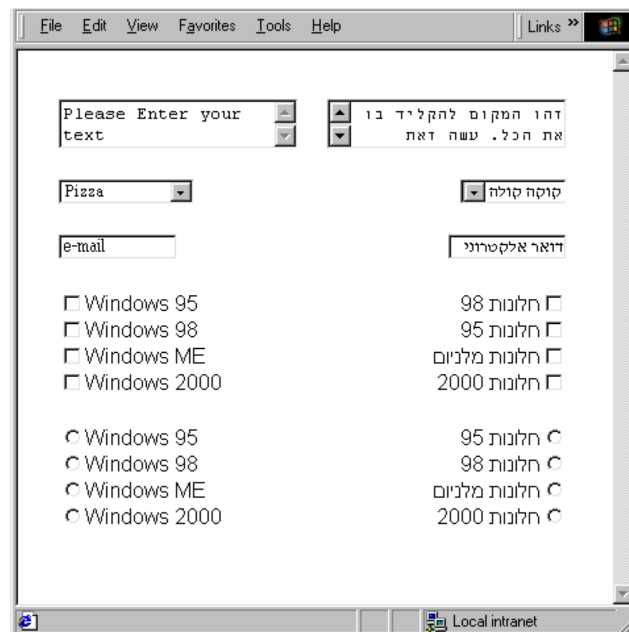
נקודה נוספת שיש להתייחס אליה בעיצוב טופס למילוי בעברית היא כיוון השדות, ובמיוחד שדות **select** ו-**textarea**. בתרשים 9.3 תוכל לראות בצד שמאל שדות באנגלית ובצד ימין שדות בעברית. הפרמטר הנחוץ הוא dir="rtl" שמשנה את מיקום הלחצנים בשדה.

לדוגמה, כך נראה שדה **select** אנגלי:

```
<select name="selectl">
    <option>Pizza</option>
    <option>Chiness</option>
    <option>Fried Chicken</option>
</select>
```

וכך נראה שדה **select** עברי:

```
<select name="selectr" dir="rtl">
     <option>קוקה קולה</option>
     <option>בירה מכבי</option>
     <option>תפוזינה</option>
</select>
```



תרשים 9.3: שדות בעברית (מימין) ושדות באנגלית (משמאל), קובץ **FormHeb.html**.

פרק 10

מסגרות

כעת נתחיל לבנות ממשקי משתמש רציניים...

מסגרות הן אחד הכלים היותר חזקים והיותר שנויים במחלוקת של דף HTML. לא קשה להוסיף מסגרות למסמך HTML, אך הן דורשות שינוי בדרך המחשבה לגבי המסמכים שלך. למרות שבמבט ראשון המסגרות נראות דומות מאוד לטבלאות, כוחן גדול בהרבה ומתבטא בכך, שבאמצעותן אתה יכול לחלק את האתר שלך כך שיציג יותר מדף HTML אחד בו-זמנית.

מה עומד מאחורי מסגרות

מסגרות מיסודן מהוות דרך נוספת לעיצוב מבנה האתר. באופן עקרוני, הטיפול במסגרות צריך להיות מוכר לך מנושא הטבלאות, אלא שעכשיו, עם מסגרות, יש כמה יתרונות בולטים: ניתן לחלק את המסך לחלקים, שכל אחד מהם מסוגל להתעדכן בנפרד ולהציע לגולש אפשרויות ממשק נוספות. אפילו שימוש פשוט במסגרות מאפשר לך לתת לאתר כלשהו כותרת קבועה, בזמן שמסמכים שונים מוצגים בחלון מתחתיה (שים לב לתרשים 10.1). ראה קובץ **VerticalFrames.html**.



תרשים 10.1: ממשק מסגרות פשוט בדפדפן Internet Explorer.

לחץ על הקישורים בצד שמאל של החלון וראה איך משתנה התוכן בצד הימני. השימוש במסגרות מקדם אותך צעד נוסף לעבר המטרה הנכספת של יצירת ממשק מושלם, בכך שאתה מספק למבקרים באתר כלים אינטואיטיבים וכלליים ביותר שניתן להציע.

מסגרות הן כלי מצוין למטרות הבאות:

- **תוכן עניינים** - על ידי יצירת תוכן עניינים לאורך או לרוחב המסך, אתה מאפשר למשתמש לעבור בין הנושאים השונים באתר, מבלי שייאלץ לחזור לתוכן העניינים בכל פעם שהוא מעוניין לשנות נושא. תוכן העניינים נמצא שם כל הזמן.
- **מרכיבי ממשק קבועים** - אתה יכול לקבע במסך תמונה או לוגו מסויים במסגרת שיישאר על המסך כחלק קבוע מהממשק, בזמן שהגולש מדפדף בדפים אחרים שבאתר.
- **תצוגת טפסים ותוצאות משופרת** - אתה יכול ליצור טופס במסגרת אחת ולהציג את תוצאותיו במסגרת אחרת.

יצירת מסגרת, התגית <frameset> והתגית </ frame>

כבר אמרנו שמסגרות דומות מאוד לטבלאות, ותתפלא שיצירתם הרבה יותר קלה (לא רק מכיון שאתה כבר מומחה בטבלאות) כי התחביר שלהם הרבה יותר פשוט וברור, ולכן כתיבת הקוד שלהם קלה. בעוד שכתיבת הקוד ליצירת טבלה (כמו גם תוכן הטבלה) נמצא כחלק בלתי נפרד מדף HTML, הרי שיצירת המסגרות נעשית בקובץ HTML נפרד, שכל תפקידו להגדיר את המסגרות - התוכן נמצא בדפים אחרים. אחת המשמעויות היא, שהדף אינו מוצג בדפדפן. מה שכן מוצג, אלה הם הדפים עליהם אותו דף מצביע.

עליך לזכור, שאותו דף המגדיר מסגרות מכיל את מבנה המסגרות ומצביע על הדף שימלא בתוכנו כל מסגרת ומסגרת. יצירתו ועדכונו של קובץ המסגרות צריכים להיעשות דרך פנקס הרשימות.

שתי תגיות שותפות למלאכת יצירת המסגרות. התגית **<frameset>** המגדירה את מבנה המסגרות, והתגית **</ frame>** המגדירה תוכן כל מסגרת ומסגרת.

בתגית **<frameset>** חייבים לציין את מספר השורות (rows) או את מספר העמודות (cols). שני מאפיינים אלה: **rows** ו-**cols** מקבלים ערכים מספריים (מספר פיקסלים) או אחוזים, או שילוב של שניהם. בדרך כלל, או שהכל מספרים או שהכל אחוזים. ניתן גם להשתמש בערך * (כוכבית). הכוכבית מציינת כי השטח שנותר (שורה או עמודה כלשהי), צריך למלא את החלל הנותר במסך. מספר השורות והעמודות נקבע לפי מספר הערכים המצוינים למאפיין rows או למאפיין cols.

הדוגמה הבאה מציגה יצירה של שתי מסגרות המסודרות בשורות, כאשר העליונה תתפוס 50 פיקסלים מהמסך והתחתונה את השאר:

```
<frameset rows="50, *">
```

אפשרות זו יכולה לשמש ליצירת מסך שבו יש גרפיקה בחלק העליון, והגודל המדויק ידוע כקטן מ-50 פיקסלים. אם התוכן במסגרת העליונה יהיה גדול מ- 50 פיקסלים, אז יופיע פס גלילה אנכי לאותה מסגרת.

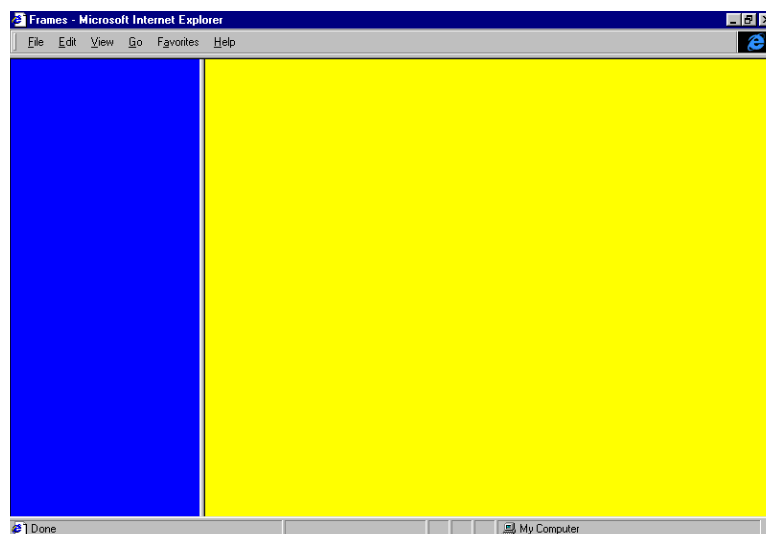
הדוגמה שלפניך יוצרת מבנה מסגרות המסודרות בשני טורים: המסגרת השמאלית תתפוס 25% מרוחב המסך, והמסגרת הימנית תתפוס 75% מרוחב המסך:

```
<frameset cols="25%, 75%">
```

אפשרות זו יכולה לשמש ליצירת ממשק בו קיים תפריט נושאים במסגרת השמאלית, ובמסגרת הימנית מופיעים המסמכים המקושרים לכל נושא מרשימת הנושאים.

בכל הגדרת <frameset> יכולה להיות הגדרת rows או cols אחת **בלבד**. כלומר, בכל הגדרת <frameset> אתה יכול להגדיר או שורות או עמודות, לא את שניהם. כדי ליצור שורות בתוך עמודות (או להיפך), אתה יכול לרשום תגית <frameset> תחת תגית <frameset> אחרת. לכל מסגרת יש תגית <frame> שתוסבר בהמשך. התחל מדוגמה פשוטה כמו זו הנמצאת בקובץ **Frame01.html**:

```
<frameset cols="25%, 75%">
   <frame src="Blue.html" />
   <frame src="Yellow.html" />
</frameset>
```



תרשים 10.2

כפי שהינך רואה, המסך מחולק לשני טורים (cols). החלק השמאלי תופס 25% משטח החלון, ואילו החלק הימני תופס 75% משטח החלון. נסה לשנות את גודל החלון (לא את המסגרת) וראה שהיחס יישמר. תמיד החלק השמאלי יתפוס 25% והחלק הימני 75%. במילים יפות, אלו הן **מסגרות דינמיות** (Dynamic Frames) כי הן משנות את גודלן בהתאם לגודל המסך ויחסית אליו.

את התגית <frame /> נסביר יותר מאוחר. עכשיו נציין שבמסגרת השמאלית יוצג דף **Blue.html** ובמסגרת הימנית יוצג הדף **Yellow.html**. להלן מבנה הדף. שים לב שכל מה שדף התוכן מציג זהו צבע בלבד.

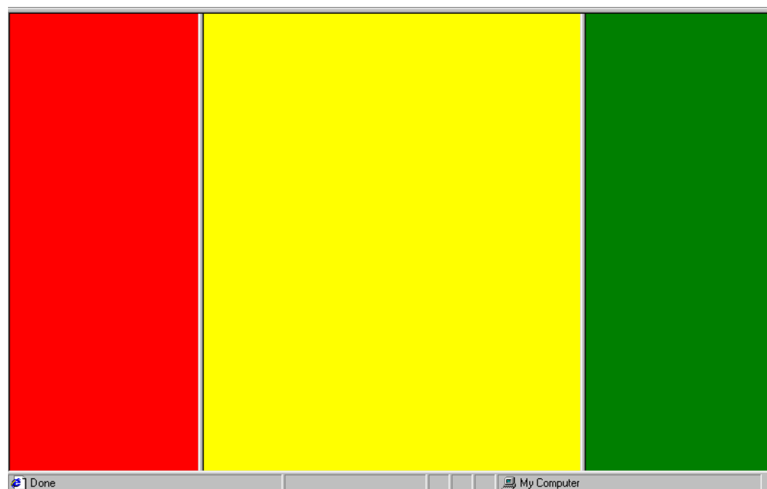
```
<body bgcolor="yellow">
</body>
```

בחלון מוצגים שני קבצים: **Blue.html** ו-**Yellow.html**. כדי לראות את התוכן של אחד מהם, תוכל להציג את הסמן על הצד השמאלי (או הימני), ללחוץ לחיצה ימנית בעכבר, לבחור **הצג מקור** (View Source) והנה לפניך הדף במלוא תפארתו. אבל, היכן הדף **Frame01.html** אותו הפעלתי? ובכן, כמו שכבר נאמר, דף זה אינו מוצג. תפקידו להגדיר את המסגרות ולציין באיזו מסגרת ייראה תוכנו של קובץ כלשהו. כדי לעדכן את קובץ **Frame01.html**, בחר בתפריט **View** באפשרות **Source**. תוכל גם לפתוח את הקובץ בעזרת **פנקס הרשימות** באמצעות **סייר Windows**. אתר את הקובץ וסמן אותו, לחץ לחיצה ימנית בעכבר ובחר **שלח אל**, **פנקס רשימות**.

תוכל לשנות את גודל המסגרות. הצב את הסמן על הגבול עד שישנה את צורתו מצורת חץ לצורת חץ דו-ראשי, לחץ לחיצה בעכבר וגרור את הגבול למיקומו החדש. התוצאה - המסגרות ישנו את גודלן. לחץ על **רענן** (Refresh) והדף ייטען מחדש עם המסגרות בגודל שנקבע בקובץ **Frame01.html**.

הנה דוגמה נוספת להצגה דינמית של מסגרות. קובץ **Frame02.html** מציג שלוש מסגרות המסודרות בטורים.

```
<frameset cols="200, 400, 200">
   <frame src="Red.html" />
   <frame src="Yellow.html" />
   <frame src="Green.html" />
</frameset>
```

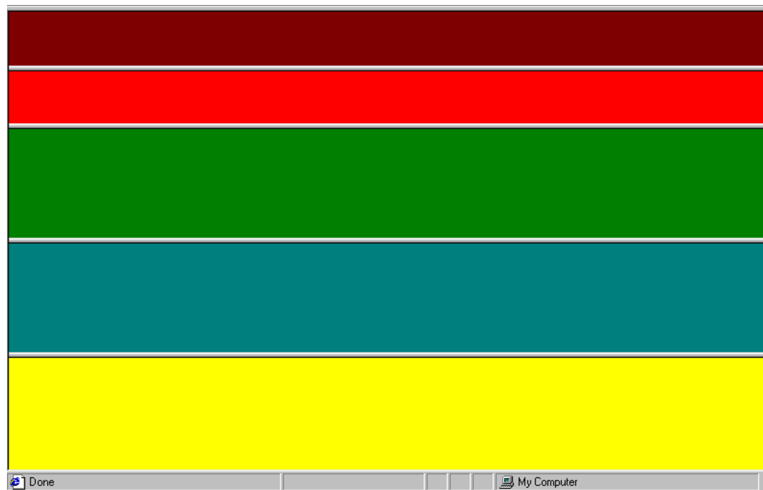


תרשים 10.3

בחלון זה מוצגות שלוש מסגרות. נתוני הפתיחה שלהן: 200 פיקסלים למסגרת השמאלית ביותר, 400 פיקסלים למסגרת האמצעית ו-200 פיקסלים למסגרת הימנית. שים לב שעם שינוי גודל החלון, יישמרו המסגרות על הגודל היחסי. במילים אחרות, המסגרת האמצעית תהיה רחבה פי שניים מהמסגרת הימנית (השמאלית), והמסגרת הימנית תהיה רחבה בדיוק כמו המסגרת השמאלית.

הדוגמה הבאה, קובץ **Frame03.html** מציג חלון עם חמש מסגרות המסודרות בשורה:

```
<frameset rows="4, 4, 8, 8, 8">
   <frame src="Maroon.html" />
   <frame src="Red.html" />
   <frame src="Green.html" />
   <frame src="Teal.html" />
   <frame src="Yellow.html" />
</frameset>
```



תרשים 10.4

בתרשים מוצגות חמש מסגרות אופקיות (מסודרות בשורה) אבל... לא לפי מה שהוגדר. לפי התגית **<frameset>** גובה המסגרת האופקית העליונה הוא ארבעה פיקסלים, גובה המסגרת מתחתיה גם הוא ארבעה פיקסלים וגובה שלוש המסגרות שמתחתיה הוא שמונה פיקסלים לכל מסגרת. במציאות, ארבעה פיקסלים זה משהו קטן מאוד ופה התקבלו מסגרות צבעוניות ו... גבוהות. ובכן, הכל בסדר. קובץ **Frame03.html** מדגים את המשמעות של מסגרות דינמיות. מה שקרה הוא, שהדפדפן חישב את כל גודל החלון והוא 4+4+8+8+8=32 ומתוכו המסגרת האופקית העליונה תופסת ארבע נקודות. ארבע נקודות מתוך 32 נקודות זה 12.5% ושמונה נקודות מתוך 32 זה 25%. במילים אחרות, ההגדרה -

```
<frameset rows="4, 4, 8, 8, 8">
```

זהה להגדרה:

```
<frameset rows="12.5%, 12.5%, 25%, 25%, 25%">
```

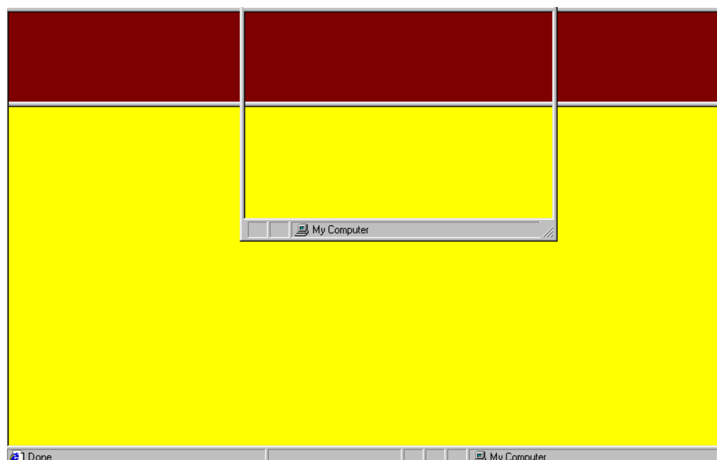
הגדרת גודל המסגרות באחוזים נעשה בקובץ **Frame04.html**.

הסימן * בין הערכים של המאפיין rows או cols מציין שאת שארית השטח יש להקצות לעמודה או לשורה.

בדוגמה הבאה מוגדרות שתי מסגרות, קובץ **Frame05.html**:

```
<frameset rows="100, *">
   <frame src="Maroon.html" />
   <frame src="Yellow.html" />
</frameset>
```

המסגרת האופקית העליונה מוגדרת בגובה 100 פיקסלים, וגובה המסגרת התחתונה יהיה שארית גובה החלון. נסה לשנות את גודל החלון ובחן את התנהגות המסגרות. שינוי רוחב החלון לא ישנה את גובה המסגרות, אבל שינוי גובה החלון ישנה רק את המסגרת התחתונה. המסגרת העליונה תישאר תמיד בגובה 100.



תרשים 10.5

זוהי נקודה חשובה, מכיון שהכנסת הכוכבית לתגית **<frameset>** שינתה את התנהגות המסגרות.

מאפיינים לתגית <frameset>

תגית **<frameset>** מגדירה את המסגרות ולה מספר מאפיינים:

מאפיין	משמעות	ערכים
cols	מגדיר את רוחב המסגרות האופקיות	pixels, %, *
rows	מגדיר את גובה המסגרות האנכיות	pixels, %, *
border	רוחב בין מסגרות	pixels
bordercolor	צבע המסגרת	blue, yellow,... #rrggbb
frameborder	רוחב בין מסגרות	0, **1**, no, yes
framespacing	רוחב בין מסגרות	pixels

הקובץ **Frame06.html** מדגים שימוש במאפיינים אלה:

```
<frameset cols="200, 400, 200" border="20">
   <frame src="Red.html" />
   <frame src="Yellow.html" />
   <frame src="Green.html" />
</frameset>
```

מאפיינים לתגית <frame />

תגית **<frame />** מגדירה את הדף שתוכנו יוצג במסגרת:

מאפיין	משמעות	ערכים
name	שם המסגרת	text
src		URL
frameborder	רוחב המסגרת	0, **1**, no, yes
marginheight	גובה שוליים של המסגרת	pixels
marginwidth	רוחב שוליים של המסגרת	pixels
noresize	לא ניתן לשנות את רוחב/גובה המסגרת	אין ערכים
scrolling	הצגת פס גלילה	yes, no, auto

קובץ **Frame07.html** מציג מסגרות, תוך כדי שימוש במאפיין **noresize** למסגרת האנכית השמאלית:

```
<frameset cols="200, 400, 200" frameborder="yes">
   <frame src="Red.html" noresize />
   <frame src="Yellow.html" />
   <frame src="Green.html" />
</frameset>
```

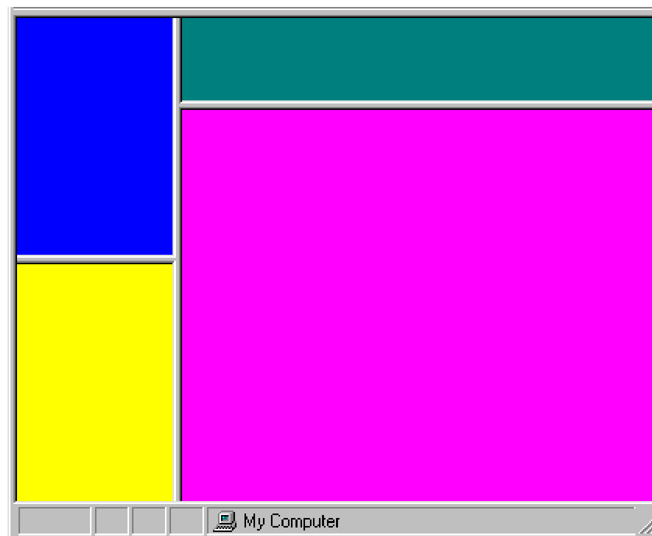
כתוצאה מצירוף המאפיין **noresize** לתגית **<frame />** לא ניתן לשנות את רוחב המסגרת. נסה לשים את הסמן על הגבול שבין המסגרת הצהובה למסגרת הירוקה. הסמן הפך מצורת חץ לצורת חץ דו-ראשי ↔. לחיצה בעכבר וגרירת הסמן ישנו את גודל המסגרות. עכשיו נסה להציב את הסמן על הגבול בין המסגרת האדומה למסגרת הצהובה. הסמן נשאר בצורת חץ ולא ניתן לשנות את רוחב המסגרות.

מסגרות מקוננות

בדומה לטבלה, נוכל ליצור בחלון מערך דו-מימדי של מסגרות. דע שזה הרבה יותר פשוט מאשר יצירת טבלה, אז קדימה לעבודה.

שורות הקוד הבאות יוצרות דף בו שתי עמודות, שבכל אחת מהן שתי שורות. ראה בקובץ **FramesRowsInCols01.html**:

```
<frameset cols="25%, 75%">
  <frameset rows="50%, 50%">
    <frame src="Blue.html" />
    <frame src="Yellow.html" />
  </frameset>
  <frameset rows="18%,*">
    <frame src="Teal.html" />
    <frame src="Fuchsia.html" />
  </frameset>
</frameset>
```

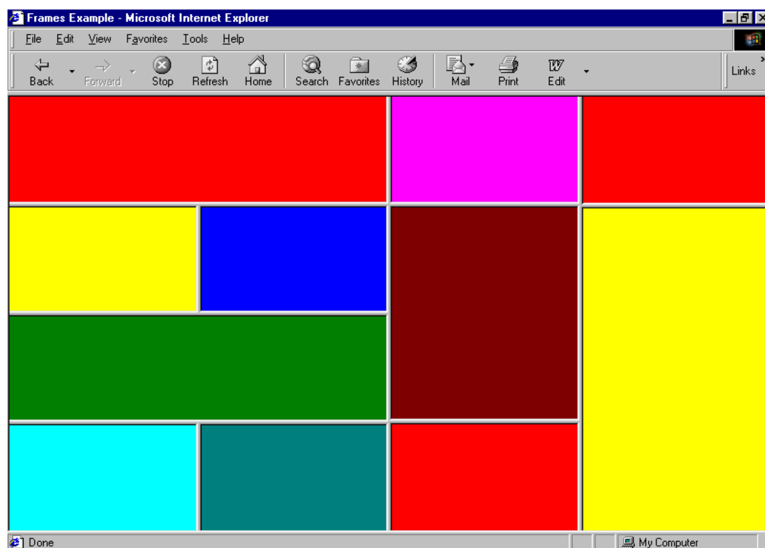


תרשים 10.6

העמודה השמאלית מחולקת לשתי שורות, שכל אחת מהן תופסת חצי (50%) מגובה החלון. העמודה הימנית מחולקת גם לשתי שורות: העליונה תופסת 18% מגובה החלון והתחתונה תופסת את יתרת הגובה שנותר. למרות שהגדרה זו בפני עצמה לא נותנת תוכן כלשהו למסגרות, היא מייצרת הפרדה ברורה של שורות ועמודות על פני המסך.

באופן תיאורטי, אין מגבלה על מספר המסגרות שניתן ליצור בחלון, אבל מבחינה מעשית יש מגבלה. כאשר מדובר בצבעים כמו בדוגמה שלעיל, אין משמעות לגודל המסגרת, אבל ככל שנגדיל את מספר המסגרות, כך יקטן גודלן ויהיה קשה יותר לקרוא ו/או לראות את תוכנן.

בקובץ **BuiltAframe.html** דוגמה מורכבת למסגרות מקוננות:



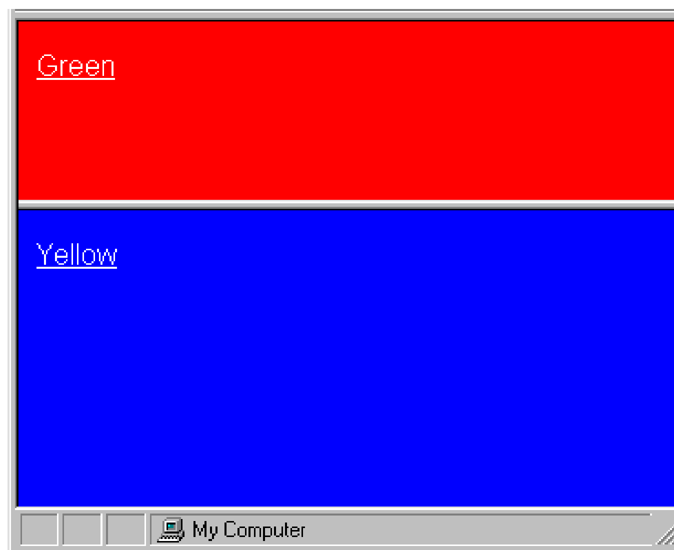
תרשים 10.7

נחזור לתגיות <frameset> מקוננות אחת בתוך השנייה בהמשך פרק זה, כשנדון ביצירת קישור בין מסגרות.

טעינת מסמכים חדשים למסגרות

עד עכשיו ראינו כיצד ליצור את מבנה המסגרות בעזרת התגיות השונות הקשורות למסגרות. חילקנו את המסך לשתי מסגרות או יותר, בהן יכולנו להציג מסמכים שונים. ברור לנו שכאן לא מסתיימת העבודה. מטרת האתר שלנו היא להציג דפים לגולש באתר, המקושרים אחד לשני. בדוגמה הבאה תוכל לראות דפים המכילים קישורים המוצגים במסגרות. קובץ **TwoFrames.html** מחלק את החלון לשתי מסגרות, כאשר בכל מסגרת יש קובץ עם קישור. הקובץ **TwoFrames.html** נראה כך:

```
<frameset rows="100, *">
   <frame src="Red2Green.html" />
   <frame src="Blue2Yellow.html" />
</frameset>
```



תרשים 10.8

על החלון יופיעו שתי מסגרות אופקיות. במסגרת העליונה יוצג הקובץ **Red2Green.html** ובמסגרת התחתונה יוצג הקובץ **Blue2Yellow.html**. לכל אחד מהקבצים האלה יש קישור אחד:

הקובץ **Red2Green.html** מכיל קישור לקובץ **Green2Red.html**. באופן מפתיע הקובץ **Green2Red.html** מכיל קישור אחד ויחיד לקובץ... נכון, **Red2Green.html**.

הקובץ **Blue2Yellow.html**, שמוצג במסגרת התחתונה, מכיל קישור לקובץ **Yellow2Blue.html**. באופן מפתיע, הקובץ **Yellow2Blue.html** מכיל קישור אחד ויחיד לקובץ... נכון, **Blue2Yellow.html**.

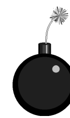
במילים אחרות, שתי המסגרות פועלות כל אחת באופן עצמאי. המסמך המקושר יופיע באותה המסגרת בה הופעל הקישור אליו.

דומני, שזהו יישום מאוד פשטני של מסגרות. הניצול היעיל של מסגרות בא לידי ביטוי בקשר שביניהן. כלומר, לחיצה על קישור במסגרת אחת והתוכן ישתנה במסגרת אחרת. בצורה זו נוכל להציג תוכן עניינים במסגרת אחת, שהפעלתו תגרום לשינוי התוכן במסגרת אחרת.

שם מסגרת

כדי ליישם קישור בין המסגרות השונות במסך, ראשית, עליך לתת לכל מסגרת שם ייחודי. ניתן לעשות זאת בעזרת המאפיין **name** של התגית **<frame />**, למשל:

```
<frame src="index.html" name="mainviewer" />
```

אזהרה **למרות שניתן לתת למסגרת כמעט כל שם, עליך להקפיד שלא להציב קו תחתון (_) בתחילת השם.** לשם המתחיל בקו תחתון יש שימוש מסוים ונתייחס אליו בהמשך.	

קישור ומסגרת

לאחר שנתת שמות למסגרות שיצרת, אתה מוכן לטעון קישור מכל מסגרת שהיא למסגרת אחרת על ידי שימוש במאפיין **target** בתגית **<a href>**, הנה כך:

```
<a href="products.html" target="mainviewer">View our Products </a>
```

ה"קישור" הוא אל מסמך שאתה רוצה שייטען למסגרת, שאת שמה ציינת במאפיין target. "שם המסגרת", צריך להיות מוגדר במסגרת עצמה באמצעות המאפיין **name** של התגית <frame />.

טיפ בכל שלב של יצירת המסמך, חשוב שניתן יהיה לקרוא אותו בקלות ושלא יידרש מהמשתמש דפדוף רב מדי בכל מסגרת. כמו כן, רצוי לבדוק את התוצאות במספר סוגי מחשבים וברזולוציות מסך שונות.	

דוגמה לשימוש בשתי מסגרות

שימוש במסגרות רבות כממשק לאתר אינו הכרחי, למרות שלא מעט אתרים עושים בכך שימוש מעניין ביותר. ניצור מבנה פשוט יותר, בו נשתמש בשתי מסגרות בלבד - האחת מציגה תוכן עניינים, ובשנייה ניתן לראות את המסמכים המקושרים לתוכן העניינים. ראה קובץ **VerticalFrames.html** להלן.

```
<frameset cols="25%, 75%">
  <frame src="store/side_menu.html" />
  <frame src="store/homepage.html" name="menu" />
</frameset>
```



תרשים 10.9

קובץ זה יוצר שתי מסגרות אנכיות. במסגרת השמאלית יוצג הקובץ **store/side_menu.html** (הקובץ **side_menu.html** בתיקיה **store**). למסגרת זו לא ניתן שם מכיון שאין צורך בטעינת מסמכים אחרים לאותה מסגרת.

menu	

במסגרת הימנית יוצג הקובץ **store/homepage.html**. למסגרת זו ניתן השם menu. אך למה ניתן שם למסגרת זו? מכיון שאנו רוצים להפעיל קישור מהמסגרת השמאלית שבעקבותיו ישתנה התוכן במסגרת הימנית, לכן אנו זקוקים ל"כתובת" המסגרת. כתובת זו היא שם המסגרת.

הנה לדוגמה אחד הקישורים מקובץ **side_menu.html**. זהו הקישור ללחצן **Office 2000**.

```
<tr>
  <td align="right">
  <a href="office_2.html" target="menu"><img src="../images/office2000.gif"
     width="120" height="31" border="0" alt="Office 2000" /></a>
  </td>
</tr>
```

לחיצה על הלחצן Office 2000 תגרום לקובץ בשם **office_2.html** (הקובץ **office_2.html** בתיקיה **store**) להיטען למסגרת ששמה menu, ובזאת הושגה המטרה.

מסגרת פנימית, התגית <iframe>

עד כה למדנו להגדיר מסגרות בעזרת התגיות <frameset> ו- <frame />. מסגרות אלו חילקו את החלון למספר חלקים. התגית **<iframe>** מגדירה **מסגרת פנימית** (Inline Frame). כלומר, המסגרת היא חלק מהדף ולא חלק מהחלון. להלן טבלה קצרה המסכמת את ההבדלים ביניהן:

	<frame />	<iframe>
הגדרה	בקובץ שאינו מוצג	בתוך דף HTML שכן מוצג
תוכן המסגרת	כל מסגרת מציגה תוכן של דף	כל מסגרת מציגה תוכן של דף

קובץ **iFrameFamily01.html** מציג מסגרת פנימית כחלק מדף html.

מסמך זה מציג טבלה עם קישורים ו**מסגרת פנימית** (inline frame) אחת. נתחיל דווקא מהגדרת המסגרת:

```
<iframe src="Family.html" name="details">
</iframe>
```

הגדרה זו מציגה מסגרת בתוך דף html ובתוכה מוצג תוכן קובץ **Family.html**. למסגרת הפנימית ניתן השם details מכיון שאנו עומדים להציג בתוכה תוכן של דפים אחרים. לשם כך, צריך "כתובת" ושם המסגרת הוא הכתובת הנדרשת. הקישורים המוגדרים במסגרת אותו דף html יציגו כל פעם דף אחר במסגרת. לדוגמה הקישור:

```
<tr>
        <td align="right">
          <a href="Amit.html" target="details">
             <font size="7">עמית</font>
          </a>
        </td>
</tr>
```

ברגע שיופעל הקישור, ייטען דף **Amit.html** במסגרת ששמה details (זהו שמה של המסגרת הפנימית).



תרשים 10.10: לחיצה על הקישור **עמית** תציג את הדף המקושר במסגרת הפנימית.

בדף זה, המסגרת וההפניה למסגרת נעשו באותו דף HTML.

המסגרת הפנימית בדף זה קטנה מדי ונמצאת שלא במקומה. לצורך מיקומה של המסגרת הפנימית נשתמש במאפיינים של התגית **<iframe>**.

מאפייני התגית <iframe>

מאפייני התגית **<iframe>** דומים מאוד למאפייני התגית **<frame />**, מכיון שאלו תגיות המגדירות מסגרות.

מאפיין	משמעות	ערכים
name	שם המסגרת	text
src	תוכן המסגרת	URL
align	יישור	**left**, center, right
height	גובה המסגרת	pixels, %
width	רוחב המסגרת	pixels, %
frameborder	הצגת מסגרת	0, **1**, no, yes
marginheight	שוליים עליונים/תחתונים	pixels, %
marginwidth	שוליים ימניים/שמאליים	pixels, %
noresize	שינוי גודל	אין ערכים
scrolling	הצגת פס גלילה	**auto**, yes, no

התבונן בקובץ **iFrameFamily02.html** שבו הוכנסה המסגרת הפנימית לתוך טבלה, שחלק קוד שלו מובא כאן:

```
<tr>
        <td align="right">
          <a href="Amit.html" target="details">
            <font size="7">עמית</font>
          </a>
        </td>
        <td rowspan="4">
          <iframe src="Family.html" name="details" width="400" height="275">
          </iframe>
        </td>
</tr>
```



תרשים 10.11

למעשה, הכנסנו את הגדרת המסגרת הפנימית לתוך תא בטבלה.

קישור בין מסגרות פנימיות

בדומה לקישור שביצענו בין מסגרות שהוגדרו תחת התגית **<frame />**, נוכל להגדיר קישור בין **מסגרות פנימיות** (inline frames) ממש באותה דרך. הכללים הם: לתת שם למסגרת, להוסיף את המאפיין **target** לקישור עם שם המסגרת שבה יוצג הדף.

חומר לימוד נוסף בנושא **HTML** בהוצאת **הוד-עמי**:

HTML 4 למפתחי אתרים באינטרנט, כ- 592 עמודים + תקליטור

חלק 3 - JavaScript

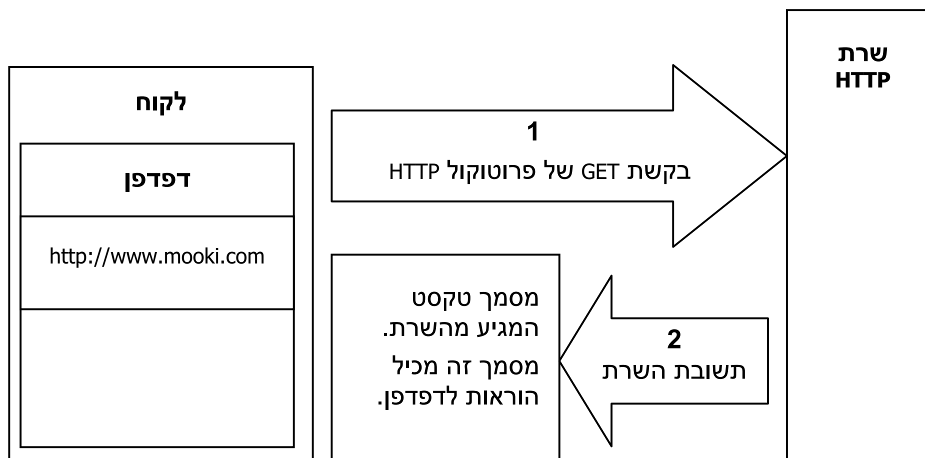
פרק 11

מבוא ל- JavaScript

כדי להבין את שפת JavaScript, יש להבין את הסביבה בה אנו מפתחים - סביבת HTTP.

בסביבת עבודה זו ישנם **שרתי HTTP ולקוחות**. השרתים הם מחשבים המחוברים לרשת האינטרנט 24 שעות ביממה וממתינים להעניק שירות ללקוחות הפונים אליהם. כאשר אנו מקלידים בדפדפן את הטקסט הזה: http://www.yahoo.com, המחשב שלנו פונה כלקוח אל שרת של yahoo ומבקש ממנו בקשת HTTP. הבקשה היא בקשת GET כלומר, המחשב שלנו, הלקוח, מבקש מהשרת לקבל דבר מה.

השרת שולח קובץ טקסט אל המחשב שלנו ובזאת ניתק הקשר. תהליך פשוט זה הינו הבסיס לכל הטכנולוגיה. התרשים הבא מתאר את תהליך הבקשה והתשובה.



תרשים 11.1

מסמך הטקסט שמגיע מהשרת מכיל למעשה הוראות לדפדפן. הדפדפן מאחסן את ההוראות בזיכרון הזמני שלו ומבצע אותן. כפי שבוודאי ניחשת, מסמך הטקסט המגיע מהשרת הוא למעשה **מסמך HTML**.

אם כך, הדפדפן מקבל מסמך HTML ומתחיל לבצע את ההוראות החל מהפינה השמאלית העליונה, תוך שהוא קורא את השורות משמאל לימין.

הדפדפן מבצע בקלות הוראות רבות, כמו זו למשל: <body bgcolor="teal">, מכיון שפקודות HTML הפשוטות אינן דורשות חישובים או הקצאת מקום בזיכרון. כיום, דפדפנים שונים יודעים לבצע פקודות נוספות הנבדלות מפקודות HTML ברמת התחכום שלהן והכוללות: הקצאת משתנים, יצירת לולאות, מערכים ועוד. כל ההוראות האלו מתבצעות על ידי הדפדפן ואינן מפעילות את השרת בפעולות חישוב כלשהן. סוג זה של תכנות נקרא **תכנות בצד הלקוח** (Client side programming).

שפת JavaScript היא אחת משפות צד הלקוח, מכיון שהיא נרשמת בתוך מסמך HTML ומתבצעת על גבי מחשב הלקוח. שפת צד לקוח נוספת היא VBScript של חברת Microsoft, אך היא מוכרת לדפדפן Microsoft בלבד (Microsoft Internet Explorer), בשעה ש-JavaScript פועלת עם דפדפני Netscape מגירסה 3 ומעלה ועל דפדפני Explorer מגירסה 4 ומעלה. יש מספר הבדלים בדרך בה פועלת JavaScript עם שני הדפדפנים.

JavaScript היא **שפה מבוססת אובייקטים** (Object Based), השונה למשל משפת ++C שהיא **שפה מכוונת עצמים** (Object Oriented). הדיון בהבדל בין Object Based ל-Object Oriented הוא מחוץ למסגרת ספר זה. תחילה, נסביר את תפיסת השימוש באובייקטים (יש הקוראים לאובייקטים בשם עצמים).

בבניית אלגוריתם, שהינו תרשים פעולה עבור תוכנית מחשב, עלינו להנחות את המחשב בכל צעד. למשל, אם נרצה לתאר נסיעה במכונית, נצטרך לתאר למחשב את המכונית על כל רכיביה ולהסביר את תהליך ההצתה ופעולת הבוכנות, את פעולת התמסורות וסיבוב הגלגלים. לאחר שנעשה זאת, נצטרך לחזור על התיאור עבור כל נסיעה או עבור כל מכונית אחרת. בדרך זו ניאלץ לכתוב שורות קוד רבות. ניתן לקצר תהליך זה על ידי שימוש באובייקטים.

אובייקט - object

אובייקט הוא תפיסה בה אנו מתארים את הסובב אותנו: מכונית, בית וכדומה כדי שהמחשב יבין ויהיה קל לנו לכתוב קוד. אפשר להתייחס לאובייקטים כאל **מכולות** (containers) של מידע. הבית מכיל מספר חלקים: סלון, מטבח, חדר שינה. אם ניקח את הסלון, לדוגמה, גם הוא מכיל מספר חלקים: טלוויזיה, כורסה, שולחן, שטיח שכל אחד מהם מורכב מחלקים נוספים עד לבורג האחרון. אם נתייחס למכונית, נראה שגם היא מורכבת מחלקים (מכלולים) בתוכה: הגה, מנוע, כסאות שכל אחד מהם הוא מכלול של חלקים אחרים.

אתה לא רק מכיר אובייקטים אלה אלא גם עובד איתם, רק שאף פעם לא קראת להם... אובייקטים. למשל, מערכת הקבצים ב-Windows היא אובייקט. אם נתייחס לרגע לדיסק C בתור אובייקט, אז הוא מכיל תיקיות (כמו Windows, My Documents ועוד). בתיקיה Windows נמצאות תיקיות נוספות כמו System, Media וכדומה, ואכן הצורה ההירכיית הזאת היא הדרך בה מתואר אובייקט למחשב.

ניתן להסתכל על אובייקטים הנמצאים בתוך אובייקטים כמו קופסה הנמצאת בתוך קופסה בתוך קופסה. התיקיה Windows היא אובייקט בתוך אובייקט הנקרא "דיסק C".

ב-JavaScript אנו משתמשים ב**תחביר הנקודה** (dot syntax), המציין שייכות של משהו למכלול ברמה גבוהה יותר. למשל, כדי לתאר את ההגה כחלק מהמכונית נכתוב CAR.WHEEL וכדי לתאר את המנוע של המכונית נכתוב CAR.ENGINE.

מאפיינים - attributes

נקרא לאובייקט המכונית בשם CAR. למכונית יש **מאפיינים** (properties): צבע, מספר דלתות, רוחב, אורך, מהירות ועוד. כל המידע הזה נמצא מוגדר באובייקט, כך שבפעם הבאה שיהיה צורך לתאר למחשב שלנו מכונית, נעשה שימוש באובייקט CAR על ידי ציון השם בלבד - CAR.

מאפיין COLOR מתאר את צבע המכונית. ב-JavaScript אנו משתמשים ב**תחביר הנקודה** (dot syntax), המציין שייכות של משהו למכלול ברמה גבוהה יותר. כלומר, כדי להתייחס למאפיין "צבע" של "מכונית", נכתוב CAR.COLOR. באותו אופן נוכל להתייחס למאפיינים של אובייקטים ברמה נמוכה יותר (נמוכה יותר מרמת האובייקט CAR). כדי להתייחס לצבע ההגה נכתוב CAR.WHEEL.COLOR וכדי להתייחס לצבע הרדיו נכתוב CAR.RADIO.COLOR.

כפי שנאמר, מערכת הקבצים במחשב היא אובייקט שיש לו מאפיינים. אם ניקח את דיסק C בתור אוביקט, אז יש לו מאפיין המתאר את שמו. כדי להתייחס למאפיין זה נכתוב c.name. כל תיקיה שבדיסק C (כלומר, באובייקט C) היא גם אובייקט ונתייחס אליה כך: C.FOLDER. גם לתיקיה יש מאפיינים כמו name ונתייחס אליהם כך: C.FOLDER.NAME. גם לקובץ יש מאפיינים כמו ReadOnly, Hidden, Size וכדומה, ונתייחס אליהם כך: C.FOLDER.FILENAME.SIZE .

שיטות - methods

לאובייקטים יש גם **שיטות** (methods). שיטות הן פעולות השייכות לאובייקט. פעולה השייכת לאובייקט המכונית (CAR) היא לדוגמה, נסיעה, אשר נציין אותה בשם DRIVE. כלומר, כשנרצה לגרום למכונית לנסוע, נצטרך לכתוב ()CAR.DRIVE.

כדי לאפיין במדויק את הפעולות המוגדרות על ידי שיטות, צריך לציין פרמטרים. נשתמש בסוגריים כדי לאפיין שיטה באופן מפורש יותר מאשר: ()CAR.DRIVE. בסוגריים אפשר לרשום ערך, למשל CAR.DRIVE(FAST). כדי לעצור את המכונית, נשתמש בשיטה ()STOP ונפעיל אותה כך: ()CAR.STOP.

באותו אופן נוכל להפעיל שיטות על אובייקטים ברמה נמוכה יותר. כדי לשנות את עוצמת הצליל ברדיו שבמכונית נפעיל את השיטה CAR.RADIO.VOLUME(LOW). כדי לפתוח את החלון הימני הקדמי נכתוב ()CAR.WINDOWS.LEFTFRONT.OPEN.

נחזור למערכת הקבצים שהיא אובייקט. יש שיטות ברמת הדיסק כמו format ויש שיטות ברמת התיקיה: העתקת תיקיה, מחיקת תיקיה, שינוי שם תיקיה. יש גם שיטות ברמת הקובץ: מחק קובץ, שנה שם קובץ וכדומה.

כעת, משהגדרנו את אובייקט המכונית ואף הגדרנו תכונות ושיטות, נוכל לפנות אל האובייקט בכל זמן. הדבר היפה בתפיסת האובייקטים הוא, שמספיק שפעם אחת נגדיר את האובייקט. לדוגמה, מספיק שפעם אחת נגדיר אובייקט בשם CAR על כל שיטותיו ומאפייניו. מעכשיו, נוכל לעשות בו שימוש לא רק בתוכנית הנוכחית, אלא בכל תוכנית הזקוקה לאותו אובייקט מסוג CAR. יהיה לנו קל מאוד לשנות את המאפיינים של האובייקט ולהפעיל את שיטותיו ויהיה לאחרים קל לגשת לאובייקט שלנו, כי הכל מובנה וברור.

ל-JavaScript יש אובייקטים מובנים ומוכנים מראש, העומדים לרשותנו ומאפשרים להעשיר את עמודי ה-HTML שלנו, תוך שימוש בשיטות ומאפיינים שונים.

פרק 12

תחביר ושורות קוד ראשונות

כתיבת קוד JavaScript במסמך HTML

הדפדפן שלנו מפרש את קוד HTML. שפת HTML מתייחסת לכל תו שאינו תחום בין סוגריים זוויתיים (< >) כאל טקסט, ומציגה אותו על המסך.

כדי להימנע מהצגת קוד JavaScript (שתחבירו אינו כולל שימוש בסוגריים זוויתיים, < >), יש לדווח לדפדפן על המעבר מ-HTML ל-JavaScript. במילים אחרות, עלינו להגדיר לדפדפן תחום מסוים בו תיכתב שפת JavaScript על ידי שימוש בתגית <script>, המבהירה כי מעתה ועד לתגית הסגירה </script> ייכתב קוד JavaScript. מכיון שמלבד JavaScript קיימת גם VBScript וגם JScript, יש להבדיל ביניהן על ידי הוספת המאפיין language, כפי שמוצג בדוגמה הבאה.

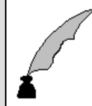
```
<script language="javascript">
JavaScript        כאן ייכתב קוד
</script>
<script language="vbscript">
VBScript        כאן ייכתב קוד
</script>
```

הסתרת קוד JavaScript מדפדפנים שאינם מכירים את השפה

כדי למנוע מצב בו דפדפן מיושן או דפדפן שאינו מכיר את התגית **<script>** ימשיך להציג על המסך את קוד JavaScript שלנו, רצוי לתחום את הקוד במסגרת הערת HTML. דפדפן המבין את התגית <script> לא יתייחס להערת HTML, מכיון שמבחינתו הוא מפרש את JavaScript. דפדפן שאינו מכיר את התגית <script> יתעלם לחלוטין מהקוד העטוף בהערה, כפי שמוצג בדוגמה הבאה.

```
<script language="javascript">
<!--
Here comes JavaScript code
//-->
</script>
```

הוספת תגית ההערה חשובה בעת פרסום האתר ב-Web, אולם הוספה זו אינה דרושה בעבודה וגם לא בעת התרגול בספר זה. לכן, אין צורך להוסיף הערה בכל תרגיל.



בעברית פשוטה!

אנו כותבים JavaScript בין תגית הפתיחה <script language="javascript"> לבין תגית הסגירה </script>.

דוגמה **12-01.html** : העתק את הקוד הבא ושמור אותו כמסמך HTML.

```
<html>
 <head>
<script language="javascript">

  alert("Welcome to Jamaica man!") ;

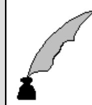
</script>
  <title>
  </title>
 </head>
</html>
```

בדוגמה זו השתמשנו באחת השיטות (method) הפופולריות של השפה, אשר מציגה חלון הודעה למשתמש. בהמשך הספר נעסוק בשיטה זו בהרחבה, ואף נשתמש בה לבדיקת הקוד על ידי הצגת הודעות במהלך ריצת התוכניות.

פתיחת דף זה על ידי הדפדפן תגרום להופעת ההודעה הבאה:



תרשים 12.1



בעברית פשוטה!

השיטה **alert()** מציגה חלון הודעה למשתמש, ועוצרת את ריצת התוכנית עד שהמשתמש לוחץ על לחצן **אישור**.

היכן כותבים את קוד JavaScript במסמך HTML?

בעיקרון, ניתן לכתוב את קוד JavaScript בכל מקום במסמך HTML. הקוד יתבצע באופן יחסי לשאר התגיות (יש לתחום את הקוד בין תגית הפתיחה **<script language="javascript">** לבין תגית הסיום **</script>**). עם זאת, בשלב מאוחר יותר, כאשר נדון בפונקציות, נכתוב את הקוד בתוך הכותרת, <head>, של המסמך.

תחביר, תחביר, תחביר

- שפת JavaScript רגישה לאותיות רישיות (גדולות) ולאותיות רגילות (קטנות). לכן, רצוי ומומלץ להקפיד על שימוש באותיות רגילות, אלא אם כן מצוין אחרת.
- בסוף משפט/פקודה בשפת JavaScript לא חובה לשים ; (נקודה-פסיק), אבל מטעמים של נוהלי תכנות תקינים אנו נקפיד על ; בסיום כל משפט.
- פעמים רבות דרושים גרשיים. ניתן להשתמש בגרש בודד או בגרשיים, אך יש להקפיד לא לבלבל בין השניים. גם אסור להשתמש באותו הסוג זה בתוך זה. בדוגמה שלפניך מוצג מה נכון ולמה לא נכון.

לא נכון:

```
alert("He was known as "The Snake" all through the land!") ;
document.write("<body bgcolor="red" text="yellow"><h1>Hello</h1>") ;
```

נכון:

```
alert("He was known as 'The Snake' all through the land!") ;
alert("He was known as " + "'The Snake'" + " all through the land!") ;
document.write("<body bgcolor='red' text='yellow'><h1>Hello</h1>") ;
```

- בדרך כלל, בכתיבת קוד דרושות הערות. את ההערות ניתן להוסיף בשתי דרכים כפי שמודגם בדוגמה זו.

```
// This is how you create a single line remark  - שני לוכסנים יוצרים הערה בת שורה
/* This is how you create a multiple row remark-
        לוכסן וכוכבית פותחים הערה מרובת שורות וכוכבית ולוכסן סוגרים אותה*/
```

פרק 13

מתחילים לכתוב

אובייקט המסמך - document

זה הזמן להכיר את אובייקט JavaScript הראשון שלנו - אובייקט המסמך, **document**. אובייקט זה מתאר את מסמך HTML החל מהתגית <body> ועד התגית </body> ולכן הוא מכיל את מאפייני המסמך, כמו צבע הרקע. מאפיין צבע הרקע יופיע ב-JavaScript כך: document.bgColor. שים לב לאות C הכתובה כאות רישית!

ניתן להשתמש בתכונות בשתי דרכים:

- הצגת ערך המאפיין:

```
<body bgcolor="blue">
<script language="javascript">
  alert(document.bgColor) ;
</script>
</body>
```

- קביעת ערך המאפיין:

```
<script language="javascript">
  document.bgColor="red" ;
</script>
```

בדרך הראשונה, ביקשנו לקבל את ערך המאפיין **bgColor** ולכן קבענו את ערך צבע הרקע בתגית <**body**> של HTML. לאחר מכן, השתמשנו בשיטה **alert()** להצגת המאפיין. שים לב, כי המאפיין נמצא בין הסוגריים ללא גרשיים!

נסה לתחום את התכונה בין גרשיים alert("document.bgColor"), ותגלה כי שימוש בגרשיים הופך את תוכן ההוראה alert למחרוזת טקסט!

בדרך השנייה, קבענו את ערך המאפיין על ידי השימוש בסימן שווה, =.

שימוש בשיטה write()

האובייקט **document** מכיל שיטה ליצירת מסמך HTML ושמה הוא **write()**. בעזרתה נוכל לחולל מסמך בדרך שמוצגת בדוגמה הבאה.

דוגמה 13-01.html :

```
<script language="javascript">
 document.write("<body bgcolor='red'  text='yellow'><h1>Hello</h1>") ;
</script>
```

ניתן אף להציג ערכי ביטויים או מאפיינים ללא שימוש בגרשיים. נעשה זאת כפי שמוצג בדוגמה הבאה.

דוגמה 13-02.html :

```
<body bgcolor="red">
  <script language="javascript">
     document.write(document.bgColor) ;
  </script>
</body>
```

נסה להריץ במחשב את הדוגמאות! התוצאה שהתקבלה - #ff0000 היא הייצוג ההקסדצימלי של הצבע red במבנה RGB.

שרשור

כאשר רוצים לחבר שני איברים שונים או יותר ב-JavaScript, כמו למשל מחרוזת טקסט ומאפיין, אנו משתמשים בסימן השרשור +.

דוגמה 13-03.html :

```
<body bgcolor="red">
  <script language="javascript">
    document.write("The value of the background color is " + document.bgColor) ;
  </script>
</body>
```

בדוגמה זו שרשרנו את מחרוזת הטקסט "The value of the background color is " למאפיין **bgColor** של האובייקט **document**. נסה להריץ במחשב את הדוגמה.

משתנים

משתנים (variables) הם תאי זיכרון השומרים ערכים. לכל משתנה יש שם המאפשר גישה אליו. לדוגמה, ניתן ליצור משתנה בשם mooki ולאחסן בתוכו את הערך 12. לאחר מכן, ניתן לבצע חישוב אריתמטי המתייחס למשתנה זה. לדוגמה: mooki + 5. התוצאה תהיה 17, מכיון שערכו של mooki בתחילת החישוב היה 12.

שמות משתנים

בכתיבת שמות יש לנהוג לפי כללים מקובלים.

- התו הראשון של שם משתנה חייב להיות אות (a..z, A..Z).
- שם יכול להיות מחרוזת תווים, החל באות אחת ועד עשרות אותיות. מותר להשתמש באותיות האלף בית הלועזיות (a..z, A..Z), בספרות (0..9) ובשני תווים מיוחדים: _ ו-$.
- אסור להשתמש ב**רווח** (space character) בשם משתנה.
- מומלץ לקבוע למשתנים שמות בעלי משמעות המתארים את מהות המשתנה ואת תפקידו, כגון: my_top_position או AccountNumber. שים לב שמקובל להשתמש במקף תחתון כדי להפריד בין מילים, מכיון שאסור להשתמש ברווח.

הגדרת משתנה

התחביר להגדרה/יצירת משתנה הוא ; var mooki. המילה השמורה var יוצרת את המשתנה, ואחריה יש לכתוב את שם המשתנה ולבסוף נקודה-פסיק (זכור שלא חובה לשים נקודה-פסיק והדפדפן יסתדר גם בלעדי תו זה, אבל אם נשים אותו, נדע מתי נגמר המשפט). תוך כדי משפט ההגדרה, ניתן גם לבצע הצבת ערך. אם נרצה להציב את המספר 12 בתוך משתנה mooki, נרשום ; var mooki = 12. הגדרה והצבה בעזרת הסימן = (שווה). כאמור, אין צורך להציב בו ערך אם לא דרוש כיון שניתן להגדיר משתנה ללא ערך התחלתי כלשהו.

שפת Javascript היא מסוג השפות הנקראות **loosely type langauge**. המשמעות בנוגע להגדרת משתנים היא שלא חובה להגדיר משתנה. בפעם הראשונה שהדפדפן ייתקל בשם משתנה, הוא יגיד לעצמו "אהה, הנה משתנה חדש!" ויגדיר אותו עבורך. בהמשך התוכנית שהוא ייתקל שוב באותו שם משתנה, הוא יאמר לעצמו: "אהה, אני כבר מכיר אותך". מטעמים של נוהל כתיבת קוד נכון, אנו נקפיד על הגדרת המשתנים במשפט var.

אפשר לכתוב:

```
var shishlik ;
shishlik = 4 ;
```

ואפשר לכתוב:

```
var shishlik = 4 ;
```

דוגמה 13-04.html:

```
<script language="javascript">
  var shishlik = 4 ;
  document.write("My dog's age is " + shishlik) ;
</script>
```

ערכי משתנים

משתנים ב-JavaScript יכולים לקבל מספר רב של סוגי ערכים. משתנים אלה נקראים **משתנים חופשיים** (free variables). לפניך פירוט של סוגי המשתנים המותרים:

- **מחרוזת טקסט (string)**: מספר תווים בין גרשיים או בין גרש בודד.

```
bellsprout = "This is my night" ;
tangela = 'This is my night ' ;
```

- **ערך מספרי**: מספר הוא כל ערך חיובי או שלילי עד 10e307! לאחר מכן, מתייחסת JavaScript למספרים כ-Infinity (אינסוף).

```
machop = -9  ;
pikachu = 10.032 ;
```

- **ערך בוליאני**: משתנה יכול לקבל ערך true (אמת, נכון, כן) או ערך false (שקר, לא נכון, לא).

```
paras = true  ;
abra = false ;
```

- **ערך ריק**: אם לא מצהירים על ערך של משתנה, הוא מקבל ערך ריק. כלומר, null. לא ניתן לבצע פעולות אריתמטיות על ערך null.

לצורך תרגול, בצע את הדוגמה הבאה המשלבת סוגי ערכים שונים:

דוגמה 13-05.html:

```
<html>
  <head>
      <script language="javascript">
        var the_person_name = "Yaron" ;
        var the_person_age = 27 ;
        var married = true ;
        document.write( the_person_name + " is " + the_person_age +
          ↳ " years old<br />") ;
        document.write( "To the question 'Is he married?', the answer is : "
          ↳ + married) ;
      </script>
    <title>
    </title>
  </head>
</html>
```

ניתן לוותר על המילה var, אך נוח יותר להבין קוד בו מציינים אותה כשמגדירים משתנה חדש.

הסימן ↳ פירושו, ששורה זו הינה המשך של השורה הקודמת ויש להקליד ברצף וללא לחיצת Enter. לחיצת Enter רק לאחר התו ;.

ניתן להגדיר מספר משתנים באותה שורה על ידי שימוש בפסיקים, כגון:

```
var bulbasaur="hello", drowzeemooki=14, eliyahoo=false ;
```

עם זאת, מומלץ להגדיר כל משתנה בשורה נפרדת לשם הסדר ובהירות הקריאה, כך:

```
var bulbasaur ;
var drowzee ;
var eliyahoo = false ;
bulbasaur = "hello" ;
drowzee = 14 ;
eliyahoo = false ;
```

פעולות עם משתנים

ניתן לבצע פעולות מתמטיות על משתנים המכילים מספרים. למשל:

```
drowzee = 14 ;
drowzee = drowzee + 3 ;
```

המשפט הראשון מציב את הערך 14 במשתנה בשם drowzee. המשפט השני הוא משפט הצבה. בצידו הימני נלקח הערך של משתנה drowzee השווה 14 ונוסף לו הערך 3. התוצאה מוצבת במשתנה drowzee "ודורסת" את הערך שהיה קיים בה קודם. כלומר, עכשיו הערך במשתנה drowzee הוא 17.

```
drowzee = "14" ;
drowzee = drowzee + 3 ;
```

האם אתה יכול לשער מה תהיה תוצאת שני המשפטים האלה? במשפט ההצבה הראשון מוצבת המחרוזת "14" במשתנה בשם drowzee. זוהי המחרוזת "14" ולא המספר 14. במשפט ההצבה השני נעשה ניסיון לבצע פעולה על מחרוזת "14" ומספר 3. התוצאה תהיה לא "+" במובן חיבור אלא "+" במובן שירשור, והתוצאה תהיה "143".

זכור, לא ניתן לבצע פעולות חיבור וחיסור על משתנה מסוג מחרוזת טקסט, אפילו אם המחרוזת מכילה מספר בלבד.

פרק 14

לולאות ובקרת זרימה

בנוסף לשימוש במשתנים, JavaScript נותנת כוח נוסף לדפי HTML על ידי הוספת לולאות ובקרת זרימה. **לולאה** היא דרך לבצע פעולה או רצף פעולות מספר פעמים מוגדר, כגון הדפסת מחרוזת טקסטואלית חמש פעמים, ללא צורך לכתוב את הפקודה יותר מפעם אחת.

ב-JavaScript מספר דרכים לכתיבת לולאה. הדרך הראשונה היא הלולאה **while**.

הלולאה while

התחביר לכתיבת הלולאה **while** הוא:

```
while(condition)
 {
 JavaScript statement
 }
```

כלומר, **כל עוד** (while) מתבצע **התנאי** (condition) שרשום בין הסוגריים, יתבצעו הוראות JavaScript שכתובות בין זוג הסוגריים המסולסלים {}. לאחר ביצוע ההוראות ייבדק התנאי שוב. אם התנאי ימשיך להתקיים, יבוצעו שוב ההוראות שבין הסוגריים המסולסלים ושוב ייבדק התנאי. כך תבוצענה ההוראות שוב ושוב, עד אשר יחדל התנאי להתקיים. במצב זה תמשיך התוכנית לאחר הסוגר המסולסל { שבסוף קבוצת ההוראות.

הבט בדוגמה הבאה:

```
<script language="javascript">
  var a = 1 ;
  while (a < 5)
    {
     document.write(a + "<br />") ;
    }
</script>
```

בדוגמה זו, נוצר משתנה בשם a שערכו המספרי 1. כל עוד ערכו של a קטן מ- 5, ערך זה יוצג על המסך. הלולאה שבדוגמה זו תפעל לנצח, כי בכל פעם שהתנאי ייבדק הוא יתקיים (ערכו true) וההוראות תבוצענה שוב.

אם רוצים להגביל את מספר הריצות של הלולאה, יש לדאוג שהתנאי יחדל להתקיים. נעשה זאת על ידי שינוי ערך משתנה a.

בצע את הדוגמה הבאה.

דוגמה 14-01.html :

```
<html>
 <head>
    <script language="javascript">
      var a ;
      a = 1 ;
      while (a < 5)
       {
        document.write(a + "<br />") ;
        a = a + 1 ;
       }
    </script>
   <title>
   </title>
 </head>
</html>
```

בדוגמה זו הוספנו בתוך תחום הלולאה **while** שבין הסוגריים המסולסלים (בין { לבין }), ביטוי המגדיל את ערך a ב- 1. לפיכך, הלולאה תפעל ארבע פעמים בלבד כי בכל ריצה יגדל ערך a, וכשערך זה יגיע ל- 5 התנאי לא יתקיים – ערכו יהיה false.

אופרטורים

אופרטורים הם סימנים שבהם אנו משתמשים לבדיקה ולקביעת ערך. לפניך רשימות חלקיות של האופרטורים הקיימים ב-JavaScript.

אופרטורים להשוואת ערכים :

== בדיקת שוויון

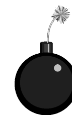
!= בדיקת אי-שוויון

>= גדול או שווה

<= קטן או שווה

> גדול מ

< קטן מ



אזהרה

אופרטור ההשוואה == וסימן השוויון = הם לא אותו דבר!

אופרטורים לוגיים:

&& וגם (צירוף של תנאים: כל התנאים צריכים להתקיים).

|| או (לפחות תנאי אחד צריך להתקיים).

ניתן לבדוק קיום של מספר תנאים בדרך זו: while (a>1 && a<10).

כלומר, כל עוד a גדול מ- 1 **וגם** a קטן מ- 10.

בדיקה של אחד ממספר תנאים תתבצע כך: while (a<1 || a>10).

כלומר, התנאי כולו מתקיים אם a קטן מ- 1 **או** גדול מ- 10.

אופרטורים מתמטיים:

+ חיבור

- חיסור

* כפל

/ חילוק

% מודולוס, המציין שארית של פעולת חילוק (לדוגמה: 23%7=2).

הצבת ערך במשתנה

בדוגמה **14-01.html** השתמשנו בביטוי a=a+1 כדי להגדיל את ערכו של a באחד. ניתן לבצע כל פעולה אריתמטית פשוטה בדרך זו, כגון a=a*19 או a=a/2. בצע את הדוגמה הבאה.

דוגמה 14-02.html:

```
<html>
 <head>
    <script language="javascript">
    var a ;
    var a_counter ;
    a = 4 ;
    a_counter = 0 ;
```

```
    while (a <= 100)
     {
       document.write(a + ",") ;
       a = a + 4 ;
       a_counter = a_counter+1 ;
     }
    document.write("<hr />There are " + a_counter + " numbers that can
     ↳ be divided by 4 between 1 & 100") ;
    </script>
   <title>
   </title>
  </head>
</html>
```

תוכנית זו מציגה את כל הכפולות של 4 בין 1 ל- 100, סופרת ומציגה אותן. לשם כך יצרנו שני משתנים: משתנה a מקבל את ערכי כפולות 4 על ידי הגדלת ערכו ב- 4 בכל ריצה של הלולאה, ומשתנה a_counter סופר את מספר הפעמים שהלולאה רצה על ידי הגדלת ערכו ב- 1 בכל ריצה של הלולאה.

תנאי הלולאה הוא, כל עוד ערכו של משתנה a קטן או שווה 100.

במהלך ריצת הלולאה מוצג ערכו של a המשורשר אל המחרוזת המכילה פסיק, כדי להפריד בין המספרים. כמו כן, עולה ערכו של a ב- 4 כדי להגיע לכפולה הבאה של 4, וכן עולה ערכו של a_counter ב- 1 כדי לספור את מספר הפעמים שהלולאה רצה. לאחר סיום ריצת הלולאה (כאשר התנאי חדל להתקיים), מודפס קו הפרדה (התגית <hr />) ולאחריו מחרוזת המשלבת את ערכו של a_counter, כדי להציג את מספר הפעמים שהלולאה רצה. הלולאה מציגה את מספר הכפולות של 4 בין 1 ל- 100.

קביעת ערך

ניתן לקבוע ערך למשתנה במספר דרכים. לדוגמה, במקום לכתוב:

```
a = a + 1 ;
```

Javascipt עושה שימוש גם באופרטורים מיוחדים הנקראים **shorthand operators**.
שים לב, כל המשפטים **באותה** שורה עושים בדיוק את אותו הדבר:

a = a + 1 ;	a += 1 ;	++a	a++
a = a – 1 ;	a -= 1 ;	--a	a--
a = a + b ;	a += b ;		
a = a – b ;	a -= b ;		
a = a * b ;	a *= b ;		
a = a / b ;	a /= b ;		
a = a % b ;	a %= b ;		



הערה

שיטת סימון זו מקובלת בשפות שונות, כמו ++C, C, Java.

דוגמה 14-03.html:

```
<script language="javascript">
     var a ;
     a = 1 ;
     while (a < 5)
      {
         document.write(a++ + "<br />") ;
      }
</script>
```

שים לב שכאשר מציגים משתנה, אפשר גם לשנות את ערכו.

בדוגמה הבאה יש שימוש ב- a++ אשר מבצע את פעולת ההוספה לפני הצגת הערך של המשתנה.

השווה בין התוצאות השונות של דוגמאות **14-03.html** ו-**14-04.html**.

דוגמה 14-04.html:

```
<script language="javascript">
      var a ;
      a = 1 ;
      while (a < 5)
       {
         document.write(++a  + "<br />") ;
       }
</script>
```

הלולאה for

הלולאה **for** היא דרך שונה ומתוחכמת יותר לביצוע לולאה. הנה התחביר שלה:

```
for (initial variable value ; condition ; variable change)
  {
     JavaScript statement(s)
  }
```

הפרמטר הראשון של ההוראה for מבטא את הערך התחילי של משתנה הלולאה, הפרמטר השני הוא התנאי לסיום הלולאה והערך השלישי הוא הערך שבו יש לקדם את משתנה הלולאה בכל מחזור.

התנאי המופרד בנקודה-פסיק פועל כמו בלולאות while. ניתן להבין זאת טוב יותר מהדוגמה הבאה.

בעת הכניסה ללולאה מוצב הערך ההתחלתי במשתנה הלולאה. תנאי הלולאה נבדק. אם הוא מתקיים, אז מבוצעים משפטי הלולאה. משתנה הלולאה מקודם, תנאי הלולאה נבדק, וחוזר חלילה כל עוד ביטוי הלולאה מתקיים.

דוגמה 14-05.html:

```
<script language="javascript">
      for(a = 1 ;  a < 5 ;  a++)
       {
        document.write(a + "<br />") ;
       }
</script>
```

שים לב שהפרמטרים של הוראת הלולאה for יכולים להיות ערכים קבועים, או משתנים ממש. אין צורך להגדיר את a לפני ביצוע הלולאה, כמו שעשינו בדוגמאות קודמות, אבל זה אפשרי לכתוב כך : for(var a=1; a<5; a++).

הפרמטר האחרון בהוראה for מציין את שינוי ערך המשתנה במהלך ביצוע הלולאה. בשל הגדרת הפעולה בתחביר הלולאה, אין צורך לציין שינוי ערך כחלק מהוראות הלולאה, כפי שעשינו בלולאה מסוג while.

לולאה בתוך לולאה - לולאות מקוננות

לעיתים יש צורך לשלב לולאות בתוך לולאות. במצבים כאלה כדאי להקפיד על כתיבה מסודרת, בה הוראות הלולאה החדשה נכתבות בהזחה מתחילת השורה.

דוגמה 14-06.html:

```
<script language="javascript">
      for (I = 1 ;  I < 10 ;  I++)
       {
        for(x = 1 ;  x < 3 ;  x++)
         {
          document.write(I + " " + x +"<br />") ;
         }
        document.write("<br />") ;
       }
</script>
```

תרגיל:

עלינו להציג את לוח הכפל 10 x 10. נעשה זאת כמובן ב- 10 שורות של 10 מספרים.

פתרון:

כשניגשים לכתיבת קוד יש להציג תחילה את אלגוריתם התוכנית. כלומר, יש לתכנן את מהלך זרימת הפעולות בתוכנית.

1. דרושה לולאה אשר תרוץ 10 פעמים עבור כל איבר בשורה, ואחריה תעבור לשורה הבאה (
).
2. דרושה לולאה אשר תדאג כי הלולאה הקודמת תפעל 10 פעמים, כדי להציג 10 שורות.

כעת, נגדיר את הלולאה המציגה שורה של 10 מספרים ונשתמש במשתנה I:

```
for (I =1 ;  I <= 10 ;  I++)
{
  document.write(??? + " ") ;  //   הכנסנו מחרוזת ריקה עבור רווח
}
document.write("<br />") ;      // לאחר הצגת 10 מספרים השורה תישבר
```

נכניס את קטע קוד זה ללולאה שתרוץ בעצמה 10 פעמים. במקום סימני השאלה נוכל לרשום את פעולת הכפל של משתנה הלולאה החיצונית בערך משתנה הלולאה הפנימית.

דוגמה 14-07.html:

```
<script language="javascript">
      var x ;
      var I  ;
      for(x = 1 ;  x <= 10 ;  x++)
       {
        for (I = 1 ;  I <=10 ;  I++)
         {
          document.write(x*I + " ") ;
         }
        document.write("<br />") ;
       }
</script>
```

1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100

Done | Local intranet

תרשים 14.1

הכנסת מספרים לטבלת HTML

למעשה, הקוד אינו שונה, מלבד פקודות הטבלה. התגיות של פתיחת הטבלה **<table>** יופיעו פעם אחת לפני תחילת הלולאות ופעם נוספת - כסגירה, לאחר סגירת הלולאה הגדולה, **</table>**. תגיות הפתיחה והסגירה של השורות, **<tr>** ו-**</tr>** יופיעו במסגרת הלולאה הגדולה ויחליפו למעשה את שבירת השורה על ידי **
. גם תגיות הפתיחה והסגירה של תא בטבלה **<td> ו-**</td>** יופיעו במסגרת הלולאות.

כמו כן, פקודות הפתיחה והסגירה של תא טבלה יופיעו כמובן בכל הצגת מספר.

דוגמה 14-08.html:

```
<script language="javascript">
     var x ;
     var I ;
     document.write("<table border='1'>") ;
     for (x = 1 ;  x <= 10 ;  x++)
      {
       document.write("<tr>") ;
       for (I = 1 ;  I <= 10 ;  I++)
        {
         document.write("<td align='right'>" + x*I +"</td>") ;
        }
       document.write("</tr>") ;
      }
     document.write("</table>") ;
</script>
```

וזו התוצאה:

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Done My Computer

תרשים 14.2

התנאי if...else

כמו בכל שפת תכנות, ניתן לבצע בדיקות נקודתיות על ידי הפקודה **if**. כלומר, ניתן לבדוק קיום של תנאים ולהתנות ביצוע הוראות מסוימות בתוצאות הבדיקה. להלן התחביר:

```
if (condition)
  {
   javascript statement
  }
```

כלומר, אם מתקיים התנאי שכתוב בין הסוגריים, יתבצעו ההוראות שבין הסוגריים המסולסלים. אם אין סוגריים מסולסלים, תתבצע רק ההוראה הראשונה שאחרי התנאי!

ניתן להוסיף גם את הפקודה **else**, אשר קובעת אלו הוראות יתבצעו כאשר התנאי אינו מתקיים. בדוגמה הבאה ניתן לראות שהתנאי הוא האם ערך המשתנה a שונה (לא שווה) ל-5. תנאי יכול לתת תשובה true או false. במקרה שהתשובה היא true, כלומר, ערכו של המשתנה a שונה מ- 5, יתבצע המשפט document.write שיציג את ערכו של המשתנה a. כאשר התנאי אינו מתקיים, כלומר התשובה היתה false, אז יתבצעו המשפט/ים מעבר למילה else. נקבל בביטוי התנאי תשובה false כאשר משתנה a לא שונה מ-5, כלומר, שמשתנה a כן שווה 5, אז תודפס המילה Five. כל הפקודות האלו נמצאות בתחום הלולאה for.

דוגמה 14-09.html:

```
<script language="javascript">
      for (a = 1 ;  a <= 10 ;  a++)
       {
        if(a != 5)
         {
          document.write(a + "<br />") ;
         }
        else
         {
          document.write("Five<br />") ;
         }
       }
</script>
```

כעת נשתמש בבדיקות if ו-else כדי לשכלל את טבלת המספרים ולהוסיף צבע אדום בכל מספר שהוא כפולה של 7. כדי לוודא שמספר הוא כפולה של 7, עליו להתחלק ב- 7 ללא שארית, כלומר (if (a%7==0, שהרי "a%7" נותן את השארית של חלוקת a ב- 7.

נכניס את התנאי לקוד הקודם, דוגמה **14-08.html**. אולם הפעם, במקום להדפיס את תא הטבלה בכל ריצה של הלולאה הפנימית, נבדוק ונחליט אם להדפיס את התא באדום או בצבע רגיל (שחור). ראה דוגמה **14-10.html**.

דוגמה 14-10.html:

```
<script language="javascript">  // open the script
      var x ;
      var I ;
      document.write("<table border='1'>") ;    // open a table
      for (x = 1 ;  x <= 10 ;  x++)                  // for each Row
       {
        document.write("<tr>") ;                     // open a Row
        for (I = 1 ;  I <= 10 ;  I++)                // for each Column
         {
          if (x*I % 7 == 0)
           {
            document.write("<td bgcolor=' khaki' align='right'>" + x*I +"</td>") ;
           }
          else
           {
            document.write("<td align='right'>" + x*I +"</td>") ;
           }
         }
         document.write("</tr>") ;                    // closing a Row
       }
      document.write("</table>") ;                  //  closing the table
</script>                                             //   closing the script
```

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Local intranet

תרשים 14.3

תרגיל:

במקום לבדוק את שארית החלוקה ב-7 בדוק את שארית החלוקה ב- 4 למשל, וראה מה קיבלת.

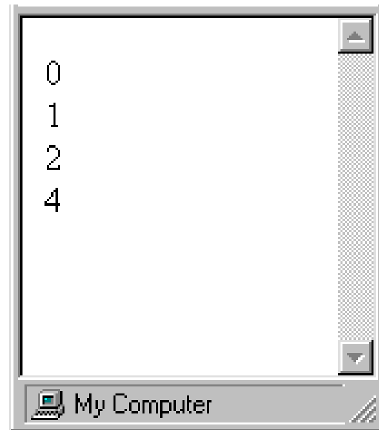
הפקודה continue

הפקודה **continue** המופיעה במהלך ביצוע הפעלת הלולאה ממשיכה את ביצוע הלולאה שוב מראש הלולאה. כלומר, היא מחזירה את סמן ריצת התוכנית אל תחילת הלולאה. בדוגמה הבאה תפעל הלולאה 5 פעמים ובכל פעם תדפיס את מספר ריצת הלולאה על ידי הדפסת משתנה הלולאה I. לפני הדפסת המשתנה I תתבצע בדיקת ערכו. אם ערכו של I שווה 3, תתבצע הפקודה continue. במקרה זה התוכנית לא תמשיך בביצוע הלולאה, אלא תדלג לראש הלולאה להמשך ביצוע. המספר 3 לא יודפס.

דוגמה 14-11.html:

```
<script language="javascript">
      var I ;
      for(I = 0 ;  I < 5 ;  I++)
       {
        if (I == 3)
          continue ;
         else
          document.write(I + "<br />") ;
      }
</script>
```

התוצאה : מספר 3 לא יודפס.



תרשים 14.4

הפקודה break

הפקודה **break** מפסיקה את ריצת לולאה. אם נשתמש בדוגמה הקודמת ונחליף את הפקודה continue בפקודה break, נכפה את הפסקת פעולת הלולאה כאשר ערכו של I יגיע ל- 3.

דוגמה 14-12.html:

```
<script language="javascript">
     var I ;
     for(I = 0 ;  I < 5 ;  I++)
      {
       if(I == 3)
         break ;
       else
         document.write(I + "<br />") ;
      }
</script>
```

בדיקת ערך בוליאני

כאשר הערך הנבדק על ידי משפט if הוא בוליאני, כלומר true או false, ניתן לוותר על ההשוואה לערכים אלה. לדוגמה, הבט בלולאת while שבדוגמה הבאה.

דוגמה 14-13.html:

```
var a ;
var TryAgain ;
a = 1 ;
TryAgain = true ;
while (TryAgain)
 {
  document.write(a + "<br />") ;
  a = a + 1 ;
  if (a < 5)
     TryAgain = true ;
    else
     TryAgain = false ;
 }
```

לפני תחילת הלולאה במשתנה TryAgain מוצב הערך true. זהו משתנה בוליאני (boolean variable).

ביטוי התנאי בלולאה הוא while(TryAgain) שהיה יכול גם להיכתב while(TryAgain == true), לפי בחירתך.

במהלך הלולאה מקודם ערכו של משתנה a ב- 1.

משפט ה-if בודק אם ערכו של a קטן מ- 5. אם ערכו של הביטוי הוא true, כלומר a אכן קטן מ- 5, אז מוצב הערך true במשתנה TryAgain כדי שהלולאה תמשיך בפעולתה. אם התשובה היא false, כלומר ערכו של a לא קטן מ- 5 (כלומר הוא גדול או שווה ל- 5), אז מוצב הערך false במשתנה TryAgain ומכיון שכך ייפסק ביצוע הלולאה.

משפט switch

שיטה חילופית לבדיקה היא שימוש בפקודה **switch**. ניתן להטיל על הפקודה switch לבדוק ערך ביטוי ולפעול בהתאם לערך שיימצא. זהו סוג מסויים של משפט if. בדוגמה הבאה נריץ לולאה מ- 0 עד 100 ונבדוק את ערך המשתנה I בדרך זו:

switch (i) {	
case 25 :	כאשר הערך במשתנה I שווה 25
break ;	
case 50 :	כאשר הערך במשתנה I שווה 50
break ;	
default:	כאשר אף משפט case לא התבצע
}	

לאחר כל משפט case נרשום את ההוראה/ות שיש לבצע ואת ההוראה break, כדי למנוע את מילוי ההוראות במקרי case העוקבים. ניתן לרשום מספר הוראות בשורות שבין משפט case אחד למשנהו, או עד סגירת הסוגריים המסולסלים התוחמים את ההוראה switch.

דוגמה 14-14.html:

```
<script language="javascript">
      for(i = 1 ;  i < 100 ;  i++)
       {
        switch(i) {
          case 25 :
             document.write("Twenty five<br />") ;
             break ;
          case 50 :
             document.write(i + "<br />") ;
         }
       }
</script>
```

פרק 15

פונקציות - התחלת עבודה אמיתית

כדי להבין את המשמעות האמיתית של הפונקציות ב-JavaScript נתרגל תחילה **פקודות משובצות/מוטבעות** (Embedded Script).

כבר הכרנו את האובייקט document ואת המאפיין bgColor שלו, המייצג את צבע הרקע של הדף. כעת, נאפשר למשתמש לשנות את צבע הרקע על ידי פקודה מוטמעת.

Trigger

Trigger - *'**הדק**'*, הוא למעשה אירוע מסוים המפעיל הוראה. לדוגמה, סיבוב המפתח ברכב הוא ה'הדק' הגורם להנעת הרכב. דוגמה קרובה יותר לתחום בו אנו עוסקים היא לחיצה על **קישור** (link) מסוים, המשמשת 'הדק' לגלישה לאתר המסומן. לדוגמה:

```
<a href="http://www.hod-ami.co.il"> go to yahoo </a>
```

קטע השורה המסומן הוא סוג של **Trigger**.

Trigger נוסף הוא מעבר עכבר מעל אובייקט בדף. זהו Trigger שנקרא **onmouseover**. Trigger זה נכתב במסגרת התגית <a> שהיא תגית קישור (ב-Internet Explorer ניתן להחיל Trigger זה גם על אובייקטים נוספים, כגון פיסקה, כפי שנסביר בהמשך).

```
<a href="http://www.hod-ami.co.il"
onmouseover="document.bgColor='red'"> point with your mouse here to change
the background color</a>
```

כלומר, ברגע שמעבירים את סמן העכבר על טקסט הקישור תתבצע שורת הקוד הקובעת את ערך התכונה bgColor.

שים לב, שהמילה red היא מחרוזת טקסט ולכן צריך להקיפה בגרשיים. עם זאת, שימוש בגרשיים יפגע בגרשיים המקיפים את המשפט כולו, ולכן יש להשתמש בגרש בודד. ניתן לראות זאת בדוגמה הבאה.

דוגמה 15-01.html:

```
<html>
 <head>
  <title>
  </title>
 </head>
 <body bgcolor="lightblue" text="yellow">
  <h1>Welcome to the color change site !</h1>
  <hr width="300" />
  <br />
  Point with your mouse
  <a href="http://www.hod-ami.co.il"
     onmouseover="document.bgColor='red'">
  here</a>
  to change the background color!
 </body>
</html>
```

Trigger נוסף הוא **onmouseout**. Trigger זה מזהה את יציאת העכבר מתחום הקישור. ניתן לצרף את בדיקת Trigger זה אל הקוד הקודם:

```
<body bgcolor="lightblue" text="yellow">
  <h1>Welcome to the color change site !</h1>
  <hr width="300" />
  <br />
   Point with your mouse
  <a href="http://www.hod-ami.co.il"
     onmouseover="document.bgColor='red'"
    onmouseout ="document.bgColor='lightblue'">
  here</a>
  to change the background color!
</body>
```

פונקציות

פונקציה היא הוראה, או מספר הוראות, בעלת תחום פעולה מוגבל בדרך כלל - ביצוע פעולה מוגדרת - והיא אינה נמצאת ברצף הפקודות של התוכנית. כדי לגרום לביצוע ההוראות הכלולות במסגרת הפונקציה, צריך להפעיל אותה על ידי פנייה מפורשת אליה, פעולת "קריאה" (**call**).

תחביר הפונקציה:

```
function yaron()
```

ההצהרה שהקוד הוא של פונקציה נעשית על ידי המילה השמורה function ואחריה שם הפונקציה וסוגריים ריקים. נשתמש בסוגריים בשלב מאוחר יותר להעברת נתונים (פרמטרים) אל הפונקציה.

את ההוראות הנכללות בפונקציה תוחמים בין סוגריים מסולסלים :

```
function  yaron()
{
document.bgColor='red'
}
```

מיקום הפונקציות בתוכנית

את הפונקציות אפשר לכתוב בכל מקום בין פקודות התוכנית, ובלבד שהן תיכתבנה בין תגית **<script>** לתגית **</script>**. עם זאת, כדי למנוע מצב בו תתבצע קריאה לפונקציה שעוד לא נטענה אל הדפדפן, רצוי מאוד לכתוב את הפונקציות במסגרת תגיות <head>.

כתוב את הקוד הבא. אל חשש! דוגמה זו לא תתן תוצאה!

דוגמה 15-02.html:

```
<html>
  <head>
      <script language="javascript">
       function mooki()
        {
         document.bgColor='red' ;
        }
      </script>
    <title>
    </title>
  </head>
  <body bgcolor="black" text="yellow">
    <h1>Welcome to the function page !</h1>
    <hr />
  </body>
</html>
```

כפי שבוודאי שמת לב, בדוגמה זו לא התבצעה הוראת שינוי לצבע הרקע. כדי להפעיל את הפונקציה ()mooki, יש לקרוא לה ולהפעיל אותה במפורש.

קריאה לפונקציה

ניתן להשתמש במספר 'הדקים' כדי לקרוא לפונקציה. בדוגמה זו נשתמש בתגית הקישור <a> ובמאפיין href. בקריאה לפונקציה צריך לציין את שם הפונקציה (והסוגריים הריקים שאחריה), אולם אם הקריאה לפונקציה נעשית דרך המאפיין herf של התגית <a>, יש להוסיף את המילה :javascript לפני השם ובצמוד לו.

הוסף לדוגמה **15-02.html** את הקישורית הבאה :

```
click <a href="javascript:mooki()"> here </a> to change the background
```

התוכנית המעודכנת נראית כמו זו שבדוגמה הבאה.

דוגמה 15-03.html:

```
<html>
 <head>
    <script language="javascript">
     function mooki()
      {
       document.bgColor='red' ;
      }
    </script>
   <title>
   </title>
 </head>
 <body bgcolor="black" text="yellow">
   <h1>Welcome to the function page !</h1>
   <hr />
   click <a href="javascript:mooki()">here</a>
     to change the background.
 </body>
</html>
```

רשימת Triggers

Trigger	מופעל באירוע
onabort	המשתמש עוצר טעינת תמונה.
onblur	האובייקט מאבד את המיקוד כאשר אובייקט אחר נבחר במקומו.
onchange	אובייקט שמאבד מיקוד נבדק, כדי לדעת אם חל שינוי בערכו ההתחלתי, ואז מופעל ההדק.
onclick	מתבצעת לחיצה על אובייקט לחיץ.
ondblclick	לחיצה כפולה, שנעשית בדפדפן מגירסה 4.
onerror	שגיאה בהרצת הקוד.
onfocus	פעולת הדק הפוכה מ-onblur. מופעל כאשר אובייקט נבחר ומקבל מיקוד.
onload	לאחר טעינת המסמך. נמצא בתחום תגיות body.
onmouseout	יציאת עכבר מתחום אובייקט.
onmouseover	מעבר עכבר על אובייקט. כדי להבטיח תמיכה בכל הדפדפנים כדאי לשתול את ההדק בתוך פקודת קישורית, למרות שבדפדפני Internet Explorer ניתן להשתמש בהדק זה גם על האובייקטים עצמם.
onmouseup onmousedown	הבחנה בין לחיצת לחצן העכבר לבין שחרורו (גירסה 4 של הדפדפנים).
onunload	יציאה מדף.

דוגמאות:

```
<img src="joe.gif" onabort="yaron()" />
<input type="text" onblur="yaron()" />
<input type="text" onfocus="yaron()" />
<input type="checkbox" onchange="yaron()" />
<input type="button" onclick="yaron()" />

onmouseover:
<a href="javascript:void(0)" onmouseover="yaron()"><img src="loo.gif" /></a>
 <img src="loo.gif" onmouseover="yaron()" />

onmouseout:
<a href="javascript:void(0)" onmouseout="yaron()"><img src="loo.gif" /></a>
 <img src="loo.gif" onmouseout="yaron()" />
```

```
<a href="javascript:void(0)" ondblclick="yaron()">hello</a>
<input type="button" onmouseup="yaron()" />
window.onerror=yaron() ;
<body onload="yaron()">
<body onunload="yaron()">
```

פונקציות מחזירות ערך

כשפונקציה מסיימת את פעולתה היא מחזירה ערך כלשהו. כלומר, השימוש בפונקציות ברוב שפות התכנות הוא כדי לבצע שינוי בערך משתנה ולקבל ערך בחזרה, או כדי לבצע פעולה כלשהי ולהחזיר ערך תוצאה. ב-JavaScript אנו בונים את רוב הקוד בפונקציות, ולכן הפונקציה מאבדת את ייעודה המקורי. עם זאת, ניתן להפעיל אותה גם לפי ייעודה המסורתי להחזרת ערך. נבקש לקבוע את שורת הסטטוס על ידי המשפט הבא:

```
window.status='2' ;
```

כתוצאה נקבל בשורת הסטטוס את המחרוזת '2'. אולם, ניתן לקבוע ערך דינמי על ידי קביעת הערך כקריאה לפונקציה. לדוגמה:

```
window.status=joe() ;
```

אז, ייקבע ערך שורת הסטטוס על פי הערך שיוחזר מהפונקציה. ניתן להשתמש ב-JavaScript במילה השמורה return. פקודה זו מסיימת את פעולת הפונקציה ו"יוצאת" ממנה, וגם מחזירה את הערך שרשום אחריה. לדוגמה:

```
function joe()
  {
    return "hello" ;
  }
```

בדוגמה זו הפונקציה תחזיר את המחרוזת "hello". בדוגמה **15-04.html** ניתן לראות שלחיצה על הלחצן קובעת את ערך שורת הסטטוס על ידי קריאה לפונקציה ()joe.

דוגמה 15-04.html:

```
<html>
  <head>
      <script language="javascript">
       function joe()
        {
         return "Hello" ;
        }
      </script>
    <title>
    </title>
  </head>
  <body>
    <form>
      <input type="button"
         value="click here and look at the status bar"
         onclick="window.status=joe()" />
    </form>
  </body>
</html>
```

בדוגמה הקודמת הפונקציה החזירה מחרוזת, אך היא יכולה גם להחזיר ערך של משתנה, כפי שניתן לראות בדוגמה הבאה.

דוגמה 15-05.html:

```
<html>
  <head>
      <script language="javascript">
       function joe()
        {
         x = 2 ;
         return x ;
        }
      </script>
    <title>
    </title>
  </head>
  <body>
    <form>
      <input type="button"
         value="click here and look at the status bar"
         onclick="window.status=joe()" />
    </form>
  </body>
</html>
```

כל אחד יכול לבחור לטעמו את דרך הקריאה לפונקציה.

פרק 16

משחקים עם תמונות, אנימציה ומערכים

כדי לבצע שינויי תמונות במסמך HTML יש צורך להתייחס לתמונה בשם המזהה אותה. חשוב להבין כי תגית התמונה ב-HTML אינה מציינת תמונה ממש, אלא **מיקום** של תמונה במסמך!

כלומר, כאשר אנו משתמשים בתגית **<img />**, למעשה אנו אומרים לדפדפן להכין מקום עבור התמונה. ניצור מקום לתמונה ונכנה אותה בשם:

```
<img name="tmuna" />
```

כעת, כאשר יש מקום לתמונה במסמך ואף שם המציין מקום זה בצורה חד-ערכית, ניתן לפנות למאפיינים של מקום התמונה. תכונות אלו מאפשרות לקבוע, לדוגמה, את הקובץ שיופיע במקום זה. הכן שתי תמונות והשתמש באחת מהן.

```
<img name="tmuna" src="1.gif" />
```

מכיון שמיקום התמונה קבוע, ניתן לפנות אליו דרך פונקציות JavaScript ולשנות את ערך המאפיין src. להזכירך, המאפיין src מציין את סוג הקובץ.

בדוגמה **16-01.html**, נבנה תוכנית המחליפה תמונה כאשר העכבר עובר מעליה. לצורך כך נכין מקום לתמונה ונמקם בו קובץ תמונה מסוים.

נוסיף את ה-**Trigger** למעבר עכבר בשם **onmouseover**.

```
<img name="tmuna" src="1.gif" border="0" onmouseover="tmuna_change()" />
```

ה-Trigger קורא לפונקציה tmuna_change אשר טרם נוצרה. בפונקציה זו נפנה אל מיקום התמונה ונשנה את ערך התכונה src. כדי לפנות למיקום תמונה בדף, יש לציין את שם מיקום התמונה (במקרה זה tmuna) שהיא אובייקט של ה**מסמך** (document), כלומר document.tmuna. לאחר ציון שם התמונה יש לקבוע את ערך התכונה src.

```
<script language="javascript">
      function tmuna_change()
       {
        document.tmuna.src="2.gif" ;
       }
</script>
```

נסה בעצמך את הדוגמה **16-01.html**.

דוגמה 16-01.html:

```
<html>
 <head>
    <script language="javascript">
     function tmuna_change()
      {
       document.tmuna.src="2.gif" ;
      }
    </script>
   <title>
   </title>
 </head>
 <body>
   <img name="tmuna" src="1.gif" border="0"
      onmouseover="tmuna_change()" />
 </body>
</html>
```

כלומר, במעבר על התמונה תתבצע קריאה לפונקציה tmuna_change() אשר קובעת ערך חדש למאפיין src של המיקום tmuna.

העברת נתונים לפונקציה

בדרך שהודגמה לעיל היה צורך לבנות פונקציה נפרדת עבור כל החלפת תמונה. זה בסדר לשתיים או שלוש החלפות, אבל מה לעשות עם יש לבצע החלפות רבות? האם צריך לבנות פונקציה עבור כל החלפה? לא. ניתן לבנות פונקציה מודולארית המקבלת נתונים מהקריאה אליה. כלומר, כל הפניות יתבצעו לפונקציה אחת שתדע לטפל במיקומים ותמונות שונות.

העברת הנתונים מתבצעת בקריאה עצמה בין הסוגריים, כלומר:

```
<img name="tmuna" src="1.gif"
   onmouseover="tmuna_change('tmuna','2.gif')" />
```

הנתונים שנבחר להעביר הם מיקום התמונה שאת המאפיין שלה ברצוננו לשנות (במקרה זה - tmuna) ואת הקובץ הגרפי שברצוננו למקם במקום זה (במקרה זה 2.gif).

כלומר, במעבר עכבר תתבצע קריאה לפונקציה tmuna_change() ויישלחו אליה שני נתונים.

על הפונקציה לקבל את הנתונים. לכן, יש להכין משתנים שיכילו את הנתונים שנשלחו. המשתנים החדשים נרשמים בתוך הסוגריים של שם הפונקציה בצורה הבאה:

```
function tmuna_change(a,b)
```

כדאי לתת למשתנים שמות פונקציונליים, כגון pic_location ו- graphic_file :

```
function tmuna_change(pic_location, graphic_file)
```

כעת, ניתן להחליף את ההוראות הקבועות בפונקציה במשתנים שיצרנו :

```
document[pic_location].src=graphic_file ;
```

שים לב לאופן הגישה לאובייקט. לא ניתן לגשת לאובייקט כך : document.pic_location.src כי לא קיים אובייקט pic_location. כדי לגשת לאובייקט בשם הערך של משתנה pic_location יש לרשום document[pic_location].src. מכיון שהאובייקט בשם tmuna הוא חלק מהאוסף (collection) של כל התמונות במסמך, ניתן לכתוב את הפונקציה גם כך :

```
document.images[pic_location].src=graphic_file ;
```

נחבר את הכל לדוגמה **16-02.html**.

דוגמה 16-02.html:

```
<html>
 <head>
    <script language="javascript">
     function tmuna_change(pic_location,graphic_file)
      {
       document[pic_location].src=graphic_file ;
      }
    </script>
   <title>
   </title>
 </head>
 <body>
   <img name="tmuna" src="1.gif" border="0"
      onmouseover="tmuna_change('tmuna','2.gif')" />
 </body>
</html>
```

כאשר ה**פונקציה מודולרית**, ניתן לפנות אליה בבקשת שינוי כל תמונה על המסך. להזכירך, המונח "מודולריות" מתייחס לקטע קוד שאינו קבוע, אלא משתנה ונקבע במהלך ריצת התוכנית. נשתמש במודולריות הפונקציה, כדי לקרוא לה גם בעת יציאת העכבר, כלומר onmouseout.

דוגמה 16-03.html:

```
<html>
 <head>
     <script language="javascript">
      function tmuna_change(pic_location,graphic_file)
       {
        document[pic_location].src = graphic_file ;
       }
     </script>
   <title>
   </title>
 </head>
 <body>
   <img name="tmuna" src="1.gif" border="0"
     onmouseover="tmuna_change('tmuna','2.gif')"
     onmouseout="tmuna_change('tmuna','1.gif')" />
 </body>
</html>
```

בדוגמה **16-03.html** מתבצעת קריאה לפונקציה ()tmuna_change גם עם יציאת העכבר מעל התמונה, אך הנתונים שנשלחים הם שונים במקצת. הנתון הראשון הוא אותו נתון, מכיון שברצוננו להחזיר את התמונה המקורית לאותו מיקום. הנתון השני קובע את הקובץ אותו ברצוננו להציג במיקום.

דרך פשוטה להחלפת תמונות במעבר עכבר

ניתן לבצע החלפה פשוטה של תמונה בעת שסמן העכבר עובר מעליה, תוך שימוש בקוד משובץ, בצורה הבאה:

דוגמה 16-04.html:

```
<img name="tmuna" src="1.gif" border="0"
     onmouseover="document.tmuna.src='2.gif'"
     onmouseout="document.tmuna.src='1.gif'" />
```

כלומר, ניתן לקבוע את ערך src של מיקום התמונה באותו מקום שבו ממוקם ה-Trigger.

אובייקט תמונה

רצוי ליצור אובייקט עבור כל תמונה דינמית שנמצאת בדף. אובייקט תמונה הוא מסוג **Image** והתחביר ליצירתו הוא:

```
yaron = new Image() ;
```

המילה new מאתחלת את האובייקט. כלומר, yaron נוצר כמשתנה אובייקט מ**סוג תמונה** ולכן ניתן לפנות למאפייניו, כלומר אל yaron.src. לדוגמה:

```
<script language="javascript">
   pic1 = new Image() ;
   pic1.src = "1.gif" ;
   pic2 = new Image() ;
   pic2.src = "2.gif" ;
</script>
```

אנימציה

כדי לייצר אפקט של אנימציה, יש להציג מספר תמונות זו אחר זו בהבדלי זמן מסוימים. הכן מספר תמונות מסוג gif או jpg לצורך ביצוע אנימציה.

אחת הדרכים הנוחות לעבוד עם מספר רב של עצמים היא דרך מערך.

מערכים

מערך הוא סוג של משתנה המכיל נתונים רבים. יש סוגים רבים של מערכים מרובי מימדים, כלומר גם הנתונים במערך מכילים נתונים נוספים. כדאי לעיין בסעיף "העברת מספר משתנה של נתונים אל פונקציה" שבהמשך פרק זה.

לפניך דוגמה למערך:

```
mooki = new Array()  ;
mooki[0] = "Hello" ;
mooki[1] = "Good bye" ;
mooki[2] = "How are you" ;
mooki[3] = "I will see you at the party" ;
```

בשורה הראשונה נוצר המערך mooki על ידי השימוש במילה השמורה new. mooki נוצר כמשתנה אובייקט מסוג Array, כלומר מערך.

לאחר מכן, מוזן ערך לתוך המערך - המערך מתחיל מ- [0].

כעת, ניתן לפנות לכל אחד מחלקי המערך ולקבל את ערכו לדוגמה: alert(mooki[1]).

ב-alert זה תופיע המחרוזת הטקסטואלית המהווה את הערך של mooki[1], כלומר "Good bye".

שלב ראשון בבניית הקוד שיריץ אנימציה הוא יצירת אובייקט התמונה. בדוגמאות בספר זה נשתמש בשלוש תמונות.

```
<script language="javascript">
      one = new Image() ;
      one.src = "1.gif" ;
      two=new Image() ;
      two.src = "2.gif" ;
      three= new Image() ;
      three.src = "3.gif" ;
```

נבנה את המערך המכיל את האובייקטים של התמונות:

```
      pic = new Array() ;
      pic[0] = one.src ;
      pic[1] = two.src ;
      pic[2] = three.src ;
```

כפי ששמת לב, עדיין לא יצרנו פונקציה ולכן כל המשתנים שיצרנו ושניצור מחוץ לפונקציות יהיו למעשה משתנים **גלובליים.** משתנים גלובליים הם משתנים ברמת הקוד, אשר ניתן להתייחס אליהם מכל פונקציה, בניגוד למשתנים **מקומיים** אשר נוצרים בתוך פונקציה וניתן לגשת אליהם רק מהפונקציה שבה נוצרו.

ניצור משתנה **גלובלי** נוסף אשר ישמש כמשתנה דינמי לצורך החלפת התמונות.

```
var a = 0 ;
```

בשלב זה ניצור את הפונקציה. הפונקציה תקבע את ערך src של מיקום התמונה tmuna, אשר ניצור בהמשך. נשתמש במערך pic ובמיקום דינמי, על ידי שימוש במשתנה a שערכו הנוכחי הוא 0.

```
function animation()
 {
   document['tmuna'].src=pic[a] ;
   a++ ;
```

שים לב כי לאחר קביעת הערך, העלינו את ערכו של a באחד. כך, בפעם הבאה שהפונקציה תופעל, יהיה ערך src של התמונה: pic[1].

כדי למנוע מצב בו יגדל ערכו של a מעל למספר האיברים (אובייקטים) במערך, נוסיף את התנאי הבא שתפקידו לחסום את הלולאה:

```
if(a > 2) a = 0 ;
```

כעת, יש לדאוג שהפונקציה תפעל שוב ושוב. נבצע זאת בעזרת השיטה **setInterval()**.

השיטה ()setInterval

לו היינו מכניסים לולאה לפונקציה, לא היינו יכולים לבצע כל פעולה אחרת במהלך הריצה של הלולאה. המשמעות היא, שלחיצה על קישורית או על לחצן, לא היתה יכולה להתקבל על ידי התוכנית. לכן, אנו משתמשים בשיטה **()setInterval**. שיטה זו יודעת להמתין פרק זמן המוגדר מראש ואז לבצע קריאה לפונקציה. כלומר:

```
setInterval('alert("joe")',1000) ;
```

הפונקציה שבדוגמה תמתין 1000 אלפיות שנייה ואז תבצע ()alert. תמתין עוד 1000 אלפיות שנייה ואז תבצע ()alert, תמתין עוד 1000 אלפיות שנייה... וכך עד שתופעל פקודת עצירה.

כדי שניתן יהיה לעצור את ההפעלה החוזרת של הפונקציה שהופעלה על ידי ()setInterval יש לאתחל את ()setInterval בתוך משתנה, כמו בדוגמה זו:

```
x = setInterval('animation()',1000) ;
```

כאשר השיטה נמצאת בתוך משתנה, ניתן לבטל אותה על ידי שימוש בשיטה **()clearInterval** בצורה הבאה:

```
clearInterval(x) ;
```

אם הפונקציה ()clearInterval תופעל על משתנה שלא הוכרז, תופיע הודעת שגיאה. כדי למנוע מצב בו תופעל הפונקציה clearInterval(x) לפני שהופעלה הפונקציה ()setInterval באמצעות משפט הצבה למשתנה x, יש להצהיר על x כמשתנה גלובלי.

כל שנותר הוא ליצור את התמונה ב-HTML על ידי שימוש בתגית </ img> ולקרוא לפונקציה ()initAnimation שבעזרת הפונקציה ()setInterval מפעילה את הפונקציה ()animation. בקוד שבהמשך, הקריאה מתבצעת על ידי Trigger בשם onload שנמצא בתגית <body>.

לפניך הקוד המלא לביצוע האנימציה.

דוגמה 16-05.html:

```
<html>
  <head>
    <script language="javascript">
     one = new Image() ;
     one.src = "1.gif" ;
     two=new Image() ;
     two.src = "2.gif" ;
     three= new Image() ;
     three.src = "3.gif" ;
```

```
    pic = new Array() ;
    pic[0] = one.src ;
    pic[1] = two.src ;
    pic[2] = three.src ;

    var a ;
    a = 0 ;
    var x ;

    function initAnimation() {
      x = setInterval('animation()',1000) ;
     }

    function animation()
     {
      document['tmuna'].src=pic[a] ;
      a++ ;
      if (a > 2) a = 0 ;
     }
   </script>
  <title>
  </title>
 </head>
 <body onload="initAnimation()">
  <img name="tmuna" src="1.gif" />
 </body>
</html>
```

מערכים והעברת מספר משתנה של נתונים אל הפונקציה

מערך (array) הוא אחד הכלים רבי העוצמה העומדים לרשות תוכניתנים ברוב שפות התכנות. המערך הוא למעשה טבלה המכילה שורות שנקראות איברים. השורה יכולה להכיל נתון אחד (עמודה אחת בטבלה) או נתונים רבים (עמודות רבות בטבלה). המערך מיועד לאחסן סדרה של נתונים, כמו: תוצאות מבחן של תלמידים בכיתה, שמות הסטודנטים בשנה א׳, רשימת פריטים שבמחסן ועוד. אם כן, במערך מאוחסנים נתונים רבים תחת שם אחד ואפשר לגשת אליהם על פי מספר סידורי (״מספר שורה״).

המערך הפשוט, שבו יש **ערך אחד** בלבד בכל איבר (עמודה אחת), הוא מערך **חד-מימדי**. לדוגמה, אם נרצה לאחסן שמות תלמידים בכיתה, נוכל ליצור מערך שלצורך הדוגמה ייקרא students. ניצור את המערך על ידי שימוש במילה השמורה new ונגדיר את המשתנה student כמשתנה מסוג מערך, על ידי שימוש במילה השמורה ()Array (שים לב לסוגריים הצמודים).

הגדרת המשתנה students כמשתנה מסוג מערך נראית כך :

```
var students ;
students = new Array() ;
```

את המערך ניתן להזין במספר דרכים. הדרך הראשונה והפשוטה היא להקליד את ערכי המערכים ישירות אל הסוגריים של המערך שלנו, Array.

לדוגמה :

```
var students ;
students = new Array("rami", "yossi", "mooki", "shooki") ;
```

כך, יקבלו הערכים את מספרם הסידורי בהתאמה לסדר הזנתם, כלומר students[2] יכיל את הערך "mooki" (השלישי ברשימה) מכיון שהמערך מתחיל מאיבר במקום 0. השימוש בסוגריים המרובעים מאפשר גישה לערכים ספציפיים במערך על פי המספר הסידורי שלהם. כדי לא להתבלבל, אני מציע להשאיר את האיבר הראשון (זה שבמקום אפס) ריק, אז נוכל לומר שהאיבר במקום הראשון נמצא באיבר סידורי מספר 1. לצורך כך נכתוב את ההצבה למערך בדרך הבאה :

```
students = new Array("", "rami", "yossi", "mooki", "shooki") ;
```

נסה את הדוגמה הבאה.

דוגמה 16-06.html:

```
<html>
  <head>
      <script language="javascript">
       var students ;
       students = new Array("", "rami", "yossi", "mooki", "shooki") ;
        document.write("The second student in the Array is " + students[2]) ;
      </script>
    <title>
    </title>
  </head>
</html>
```

הדרך השנייה להזנת ערכים אל מערך, היא על ידי קביעתם עבור כל איבר לפי מספרו הסידורי. כלומר, לאחר הגדרת המשתנה students כמשתנה מסוג מערך, נגדיר כל אחד מהערכים על פי מספר האיבר שלו, כגון :

```
students[0] = "rami" ;
```

הדוגמה הבאה יוצרת בדרך שונה את המערך שראינו בדוגמה **16-06.html**.

דוגמה 16-07.html:

```
<script language="javascript">
      var students ;
      students = new Array() ;
      students[0] = "" ;
      students[1] = "rami" ;
      students[2] = "yossi" ;
      students[3] = "mooki" ;
      students[4] = "shooki" ;
      document.write("The second student in the Array is "+students[2]) ;
</script>
```

יתרונות המערך

כאשר פועלים על נתונים במערך, ניתן להשתמש במספר הסידורי של האיברים כדי להריץ לולאות על כל האיברים או על חלק מהם, כפי שדרוש. בדוגמה הבאה נבדוק אם התלמיד shooki נמצא בין תלמידי הכיתה. מספר הפעמים שהלולאה תפעל יהיה כמספר איברי המערך - 4 פעמים בדוגמה זו. האיבר הראשון הוא מספר 1 והאיבר האחרון הוא מספר האיברים במערך. מספר איברי המערך נמצא במאפיין **length**, כלומר students.length שערכו בדוגמה זו הוא 4, מכיון שבמערך students קיימים ארבעה איברים. תחביר הלולאה ייראה כך:

```
for (I = 0 ;  I <= students.length ;  I++)
```

בכל ריצה של הלולאה נבדוק את הערך התורן במערך על פי המספר הסידורי I, אשר ישתנה במהלך ביצוע התוכנית מ- 0 עד 3. פקודת הבדיקה היא:

```
if (students[I] == "shooki")
```

חשוב לציין שערכי מערכים יכולים להיות **מחרוזות**, **מספרים** או **ערכים בוליאנים** (true/false). בתוכנית הבאה מתבצעת בדיקה פשוטה לצורך הדגמה של השימוש במספר הסידורי של איברי המערך. זוהי שיטה יעילה על מערך קטן ולא ממוין ולכן לא מומלץ להשתמש בלוגיקה פשוטה זו בתוכניות אמיתיות.

דוגמה 16-08.html:

```
<script language="javascript">
      var students ;
      students = new Array() ;
      students[0] = "" ;
      students[1] = "rami" ;
      students[2] = "yossi" ;
      students[3] = "mooki" ;
      students[4] = "shooki" ;
      for (I = 0 ;  I <= students.length ;  I++)
```

```
        {
         if (students[I] == "shooki")
           {
             document.write("I found shooki at " + I) ;
             break ;
           }
        }
</script>
```

שיטות הקשורות למערכים

- **join()** - השיטה join() מאפשרת צירוף ערכי המערך למחרוזת אחת ארוכה. כלומר, אם נשתמש במערך students ונרצה להציג את כל ערכיו ללא שימוש בלולאה נכתוב:

```
document.write(students.join()) ;
```

על המסך יוצג:

```
,rami,yossi,mooki,shooki
```

שים לב: התו הראשון שיוצג באיברי המערך הוא פסיק (,) כי קבענו שהאיבר הראשון במערך (זה שמיקומו 0) יכיל מחרוזת ריקה.

- **sort()** - השיטה sort() משמשת למיון ערכי המערך. אם ערכי המערך הם מחרוזות טקסט, השיטה sort() תסדר את איברי המערך לפי סדר אלפביתי. כלומר, אם ניצור את המערך x בצורה הזו:

```
var x ;
x = new Array("c", "a", "b", "e", "d") ;
```

נוכל לסדר את האיברים לפי סדר אלפביתי על ידי שימוש בשיטה sort(), הנה כך:

```
x.sort() ;
document.write(x.join()) ;
```

על המסך תופיע המחרוזת: a,b,c,d,e

אם איברי המערך היו מכילים מספרים, התוצאה היתה סידור של הערכים בסדר מיון מספרי עולה.

- **reverse()** - השיטה reverse() הופכת את סדר המערך. כלומר, מערך המכיל את הערכים x,a,k לפי סדר זה יהפוך לאחר שימוש בשיטה reverse() למערך המכיל את אותם ערכים בסדר הפוך: k,a,x.

מערכים רב-מימדיים

במערך רגיל יש ערך אחד בכל אחד מאיברי המערך. ניתן להעצים את המערך על ידי יצירת מערך רב-מימדי. מערך דו-מימדי כזה דומה לטבלה שבה יש עמודות אחדות. בכל אחת מהעמודות יש ערכים מאותו סוג. לדוגמה, כדי לייצג לוח שחמט נצטרך לבנות מערך דו-מימדי שיכיל את כל המשבצות בלוח. מערך המכיל את הפרטים האישיים של כל העובדים גם הוא מערך דו-מימדי ששורותיו מייצגות את העובדים השונים, וכותרתו מייצגת את הפרטים שאנו שומרים עבור כל אחד מהם.

כדי להבין את תפיסת ריבוי המימדים במערך, ניצור מערך דו-מימדי פשוט אשר יוצג בטבלה הבאה. הטבלה מייצגת לדוגמה, מקומות ישיבה באולם קולנוע שבו השורות מסומנות באותיות והמושבים בכל שורה מסומנים במספרים.

A1	A2	A3
B1	B2	B3
C1	C2	C3

בשלב זה ניצור את המערך ידנית. אחרי הכרזת X כמשתנה מסוג מערך, עלינו להכריז על כל אחד מאיבריו (במקרה זה X[0], X[1] ו- X[2]) כמערך בפני עצמו. לאחר כל הכרזה, נאתחל את ערכי איברי המערך הפנימי.

```
X = new Array() ;

 X[0] = new Array() ;
X[0][0] = "A1" ;
X[0][1] = "A2" ;
X[0][2] = "A3" ;

 X[1] = new Array() ;
X[1][0] = "B1" ;
X[1][1] = "B2" ;
X[1][2] = "B3" ;

 X[2] = new Array() ;
X[2][0] = "C1" ;
X[2][1] = "C2" ;
X[2][2] = "C3" ;
```

לאחר הגדרת המערך ואתחול איבריו בערכים, ניצור לולאה שתבנה טבלת HTML בעלת שלוש שורות ושלושה תאים בכל שורה. מכיון שטבלה היא למעשה ייצוג של מערך דו-מימדי, נציג את המערך X בתוך תאי הטבלה. לצורך כך, נבנה שתי לולאות מקוננות, אשר הראשונה תבצע שלוש ריצות והמונה שלה I ייצג את המימד הראשון של המערך, כלומר X[I]. לולאה זו גם תתחיל ותסיים את שורות הטבלה. הלולאה הפנימית תרוץ גם היא שלוש פעמים, תבנה את תאי הטבלה ותציג בכל תא את אחד מאיברי המערך. מונה הלולאה הפנימית U ייצג את איברי המימד השני, כלומר תת-המערכים X[I][U]:

```
  for (I = 0 ;  I < 3 ;  I++)
   {
    document.write('<tr>') ;
    for(U = 0 ;  U < 3 ;  U++)
     {
      document.write('<td>'+ X[I][U] + '</td>');
     }
    document.write('</tr>')
   }
```

כמובן, יש לפתוח ולסגור את הטבלה בצורה מסודרת. לפניך הקוד המלא.

דוגמה 16-09.html:

```
<html>
 <head>
   <script language="javascript">
    X = new Array() ;

     X[0] = new Array() ;
    X[0][0] = "A1" ;
    X[0][1] = "A2" ;
    X[0][2] = "A3" ;

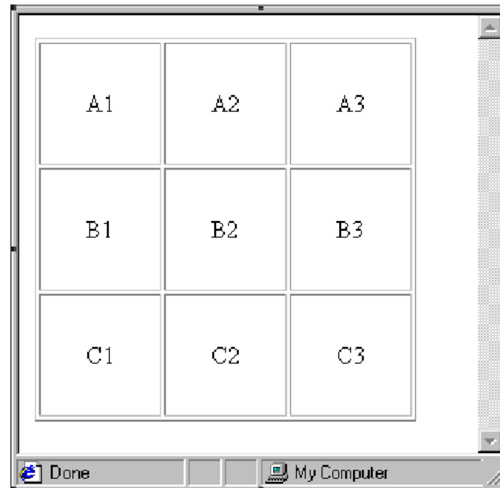
     X[1] = new Array() ;
    X[1][0] = "B1" ;
    X[1][1] = "B2" ;
    X[1][2] = "B3" ;

     X[2] = new Array() ;
    X[2][0] = "C1" ;
    X[2][1] = "C2" ;
    X[2][2] = "C3" ;

     document.write('<table border="1" cellpadding="30">') ;

     for (I = 0 ;  I < 3 ;  I++)
     {
      document.write('<tr>') ;
      for(U = 0 ;  U < 3 ;  U++)
       {
        document.write('<td>'+ X[I][U] + '</td>');
       }
      document.write('</tr>')
     }
    document.write('</table>')
   </script>
  <title>
  </title>
 </head>
</html>
```

זו תהיה התוצאה בדפדפן:



תרשים 16.1

הדוגמה הבאה משתמשת בשתי הלולאות המקוננות ליצירת המערך הדו-מימדי ולהצגתו בטבלה. התוצאה הסופית היא מבנה לוח שחמט. אם נתקלת בקשיים בהבנת הקוד, כדאי לעקוב אחר ערכי המשתנים עבור כל ריצה של הלולאה.

דוגמה 16-10.html:

```
<html>
 <head>
    <script language="javascript">
     var x ;
     document.write('<table width="100%" height="100%" border="1">') ;
     chess_board = new Array() ;
     for(I = 0 ;  I < 8 ;  I++)
       {
        x = I % 2 ;
        chess_board[I] = new Array() ;
        document.write('<tr>') ;
        for(U = 0 ;  U < 8 ;  U++)
          {
           x++ ;
           if (x % 2 == 0)
            c = 'tan' ;
           else
            c = 'white' ;
           chess_board[I][U] = I + "-" + U ;
           document.write('<td bgcolor=' + c + ' align="center">' +
              ↳ (chess_board[I][U]) + '</td>') ;
          }
```

```
        document.write('</tr>')
       }
      document.write('</table>')
     </script>
   <title>
   </title>
  </head>
</html>
```

התוצאה בדפדפן תיראה כך :

0-0	0-1	0-2	0-3	0-4	0-5	0-6	0-7
1-0	1-1	1-2	1-3	1-4	1-5	1-6	1-7
2-0	2-1	2-2	2-3	2-4	2-5	2-6	2-7
3-0	3-1	3-2	3-3	3-4	3-5	3-6	3-7
4-0	4-1	4-2	4-3	4-4	4-5	4-6	4-7
5-0	5-1	5-2	5-3	5-4	5-5	5-6	5-7
6-0	6-1	6-2	6-3	6-4	6-5	6-6	6-7
7-0	7-1	7-2	7-3	7-4	7-5	7-6	7-7

Done
My Computer

תרשים 16.2

העברת מספר משתנה של נתונים אל פונקציה

בהגדרת פונקציה אנו מציינים את מספר ה**משתנים** (arguments) שאנו מעבירים אליה, ולכן נראה שאנו חייבים לשלוח תמיד את אותו מספר משתנים. עם זאת, במציאות עלולים להיווצר מצבים בהם ניאלץ לשלוח מספר משתנים שונה בכל קריאה לפונקציה (פחות מהמקסימום האפשרי). כאשר רמת ההידודיות (אינטראקטיביות) גבוהה ועל האתר להתמודד עם החלטות המשתמש, עלינו להיערך למצבים דינמיים ולבנות פונקציות מודולריות שיכולות לטפל במספר בלתי קבוע של משתנים המועברים אליהן. ל-JavaScript יש את הכלים להתמודד עם צורך זה.

השלב הראשון בהגדרת פונקציה מודולרית שיכולה לקבל מספר משתנה של ערכים, הוא כמובן הכרזת הפונקציה. נכריז על הפונקציה joe שפועלת עם משתנה אחד.

```
function joe(x)
```

כעת נשתמש במאפיין arguments של האובייקט function. למעשה, הנתונים הנשלחים אל הפונקציה נמצאים באיברי מערך ששמו arguments. לכן, ניתן לגשת אל כל אחד מהמשתנים בנפרד. הגישה אל המערך יכולה להתבצע רק לאחר הכרזת הפונקציה.

השלב הבא הוא בניית לולאה שתבודד ותדפיס בנפרד כל נתון שנשלח אל הפונקציה. הלולאה תרוץ מהאיבר הראשון (0) ועד לסוף המערך, שכינויו joe.arguments.length.

```
for(I = 0 ;  I < joe.arguments.length ;  I++)
{
  document.write(joe.arguments[I] + ",") ;
}
```

נוסיף קריאה לפונקציה ונשלח מספר רב של נתונים, כדי לבחון את דרך הטיפול של הפונקציה במשתנים.

```
joe(1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 77, 654, 345, 788, 999, 121212, 222) ;
```

כך ייראה קוד הדוגמה :

```
<script language="javascript">

function joe(x)
 {
   for ( i = 0 ;  i < joe.arguments.length ;  i++)
    {
      document.write(joe.arguments[i] + ",") ;
    }
 }

joe(1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 77, 654, 345, 788, 999, 121212, 222) ;

</script>
```

בקוד זה אין מודולריות מכיון שתמיד נשלח אל הפונקציה את אותו מספר ערכים. כדאי לשלב תפיסה זו במצבים בהם המשתמש ממלא טפסים שנבדקים בצד הלקוח על ידי JavaScript, מכיון שאין צורך לכפות עליו למלא את כל השדות או לבנות פונקציות נפרדות המטפלות בנתונים שונים. על ידי פונקציה מודולרית ושימוש בתכונת arguments ניתן לפעול עם כל מספר של נתונים, עד למספר המקסימלי המוגדר.

פרק 17

בדיקת טקסט - אובייקט המחרוזת

אחד היתרונות הגדולים של JavaScript הוא היכולת לבצע בדיקה ועיבוד קלט המוזן על ידי המשתמש, ללא צורך לבצע גישה לשרת. כדי לטפל בטקסט אנו משתמשים באובייקט **String**.

אובייקט המחרוזת - String Object

למדנו שכדי ליצור אובייקט תמונה יש להשתמש במילה השמורה new ולציין את סוג האובייקט **Image**. כדי ליצור משתנה המכיל מחרוזת טקסט צריך רק לאתחל משתנה בעל ערך של מחרוזת טקסט. כעת, ניתן ליהנות גם מהשיטות והתכונות הכלולות באובייקט **String**.

ניצור משתנה ונעניק לו ערך של מחרוזת טקסט.

```
a = "Welcome to Jamaica" ;
```

מבחינת הדפדפן, המשתנה a הוא אובייקט מחרוזת, ולכן ניתן להשתמש בתכונה length של אובייקט המחרוזת. התכונה length מחזירה כמובן את אורך המחרוזת שהוא מספר התווים המופיעים בה.

```
a = "Welcome to Jamaica" ;
document.write (a.length) ;
alert (a.length) ;
```

כאן השתמשנו במאפיין length של האובייקט a, כדי להציג את המחרוזת על ידי document.write() וגם על ידי alert(). לביטוי a.length יש ערך מתמטי (במקרה זה 18) ולכן ניתן לבצע בו חישובים אריתמטיים, למשל (a.length + 22).

בדיקה ראשונית לטקסט שהוזן על ידי המשתמש

כדי לאפשר למשתמש להזין נתונים בדף HTML, אנו משתמשים ב**טפסים** (forms). כדי לאפשר לשפת JavaScript לגשת אל שדות הקלט, עלינו להגדיר שמות לטופס ולשדות שכלולים בו.

לדוגמה, נוכל לעשות זאת כך :

```
<body>
<form name="roller">
  Type in your personal details <hr />
  Name : <input type="text" name="first_name" /><br />
  Last Name : <input type="text" name="last_name" /> <hr />
  Click to check which of the names is longer
  <input type="button" value="click me" onclick="joe()" />
</form>
</body>
```

בדוגמה זו יש שני שדות טקסט ולחצן המכיל את ההדק **onclick**, אשר בעת לחיצה על הלחצן יפעיל את הפונקציה ()joe. אנו רוצים לבדוק ולקבוע אילו מהשמות (שם פרטי ושם משפחה) ארוך יותר. לצורך כך, על הפונקציה לברר את הערך שהוקלד אל תוך השדות. זיהוי השדה נעשה כך :

```
document.form_name.text_field_name
```

בדוגמה שלנו:

```
document.roller.first_name    וכן    document.roller.last_name
```

כלומר, אובייקט השדה הוא תת-אובייקט של אובייקט הטופס, וזה בתורו מהווה תת-אובייקט לאובייקט המסמך. אין צורך להוסיף את אובייקט ה**מסמך** (document), אולם כדאי לעשות זאת מטעמים של נוהל כתיבה נכון.

כעת נבדוק את ערך השדה ונתייחס למאפיין value שלו :

```
document.roller.first_name.value
```

נוכל להכניס את ערך השדה אל משתנה, וכך ליצור אובייקט מחרוזת :

```
        first_name_str = document.roller.first_name.value ;
        last_name_str  = document.roller.last_name.value ;
```

המשתנים first_name_str ו-last_name_str מכילים את ערכי שדות הטקסט, ולכן הם אובייקטים מסוג **אובייקט מחרוזת** (String Object). כך ניתן לבדוק את מאפיין length שלהם וגם להשוות בין שניהם, למשל.

```
     function joe()
       {
        first_name_str = document.roller.first_name.value ;
        last_name_str  = document.roller.last_name.value ;
```

```
        if (first_name_str.length != last_name_str.length)
          {
            if (first_name_str.length > last_name_str.length)
              alert("Your first name is longer and contains "
                + first_name_str.length + " characters") ;
            else
              alert("Your last name is longer and contains "
                + last_name_str.length + " characters") ;
          } else
              alert("Your first name and your last name have the same lenght: "
                  + last_name_str.length + " character(s)") ;
        }
```

ראה קובץ **17-01.html**.

שינוי התוכן של שדה טקסט

שדה טקסט הוא למעשה אובייקט ולכן ניתן לגשת למאפיין ה**ערך** שלו (value). כדי לבטא את אובייקט השדה, אנו כותבים את כל ההיררכיה עד אליו, החל מאובייקט document, דרך אובייקט הטופס ועד אובייקט השדה. למשל, כדי להתייחס לשדה הטקסט שבדוגמה הבאה:

```
<form name="former">
     <input type="text" name="texi" />
</form>
```

הרי זה כאילו כתבנו document.former.texi. כך למעשה נייצג את אובייקט השדה. כדי להתייחס אל תכונת הערך ולשנות את הכתוב בשדה, נכתוב:

```
document.former.texi.value=
```

בדוגמה הבאה נבדוק את הטקסט המוזן על ידי המשתמש. אם טקסט זה הוא "Hello", יוצב בשדה הטקסט הערך "Hello to you too". אם הטקסט שמוזן על ידי המשתמש שונה, יוצב בשדה הערך "Did you say anything?".

דוגמה 17-02.html:

```
<html>
  <head>
      <script language="javascript">
       function pulke()
        {
         if(document.former.texi.value == "Hello")
          document.former.texi.value="Hello to you too" ;
         else
          document.former.texi.value="Did you say anything?" ;
        }
      </script>
```

```
    <title>
    </title>
  </head>
  <body>
    <form name="former">
      <input type="text" name="texi" />
      <input type="button" value="Go!" onclick="pulke()" />
    </form>
  </body>
</html>
```

שיטות נוספות להצגת אובייקט מחרוזת

השיטה ()indexOf

אחת השיטות השימושיות והנוחות להצגת אובייקט מחרוזת היא **()indexOf**. בעזרתה ניתן לחפש מילים או אותיות בודדות בתוך מחרוזת. ניצור אובייקט מחרוזת ונפעיל עליו את השיטה ()indexOf, באופן הבא:

```
a ="Hello" ;
document.write(a.indexOf("o")) ;
```

התוצאה שנקבל היא 4, מכיון שהאות o (האות הקטנה) שחיפשנו במחרוזת מופיעה בה במקום הרביעי (מחרוזות ב-JavaScript מתחילות את ספירת האותיות באפס). אם הטקסט שחיפשנו אינו מופיע במחרוזת, התוצאה תהיה 1-. שים לב שהמילה Hello מכילה 5 תווים המסומנים מ- 0 עד 4.

הנה דוגמה לאימות הקלדת טקסט של דואר אלקטרוני על ידי שימוש בשיטה ()indexOf למציאת הסימן @. אם השיטה מחזירה 1-, זהו סימן שהמחרוזת אינה מכילה את הסימן, כאשר בכל מקרה אחר הסימן חייב להופיע.

דוגמה 17-03.html:

```
<html>
  <head>
      <script language="javascript">
       function email_checker()
        {
         if (document.former.texi.value.indexOf("@") == -1)
          document.former.texi.value="You forgot the Shtrudel my dear!" ;
         else
          document.former.texi.value = "Very good" ;
        }
      </script>
    <title>
    </title>
  </head>
```

```
  <body>
    <form name="former">
     e-mail:
      <input type="text" name="texi" size="40" />
      <input type="button" value="Go!" onclick="email_checker()" />
    </form>
  </body>
</html>
```

השיטה ()lastIndexOf

שיטה זו מחזירה את מיקום האות האחרונה בתת-המחרוזת המבוקשת.

```
Machop = "Designing Web Usability" ;
eIndx = Machop.indexOf("e") ;            // equal  1
eLastIndx = Machop.lastIndexOf("e") ;  // equal 11
```

השיטה ()split

השיטה **()split** יודעת לפרק מחרוזת למערך. לדוגמה, ניקח את המחרוזת הזו:

```
a = "hello,bye,mooshon,barboonia" ;
```

נוכל לפרק אותה על פי הפסיקים, בצורה הבאה: (",")a.split. החלוקה בונה מערך. הגישה לאיברי המערך מתבצעת כך:

a.split(",")[0] ייתן את הערך "hello",

a.split(",")[1] ייתן את הערך "bye",

a.split(",")[2] ייתן את הערך "mooshon" וכדומה.

השיטה ()substring

השיטה **()substring** גוזרת חלק ממחרוזת. ניצור אובייקט מחרוזת ונחיל עליו את השיטה ()substring:

```
a = "encyclopedia" ;
shellder = a.substring(4, 8) ;  //  equal  clop
```

נקבל את תת-המחרוזת "clop", מכיון שתת-המחרוזת תיגזר החל מהאות הרביעית ועד האות השמינית (לא כולל).

השיטה charAt()

השיטה **charAt()** מחזירה תו אחד מהמחרוזת לפי פרמטר שנשלח אליה.

```
a = "encyclopedia" ;
shellder = a.substring(3, 4) ;  //  equal  y
diglett = a.charAt(3) ;           // equal y
```

כעת נראה דוגמה לשימוש בשתי השיטות sunstring() ו-charAt(), כדי לגלול מחרוזת תווים. הלוגיקה שבגלילת טקסט פשוטה למדי. נדגים זאת בעזרת המחרוזת "abcd". מכיון שהטקסט צריך לנוע שמאלה, האלגוריתם יגזור את האות הראשונה ויציב אותה בסוף המחרוזת:

abcd

bcda

cdab

dabc

abcd

כלומר, האלגוריתם מרכיב מחרוזת חדשה משתי תת-מחרוזות. המחרוזת הראשונה מורכבת מהאות השנייה (ב-JavaScript זוהי האות מספר 1, כי ספירת התווים במחרוזת מתחילה ב- 0) ועד סוף המחרוזת (את סוף המחרוזת נייצג כ-a.length). למחרוזת זו נוסיף את המחרוזת השנייה, המורכבת מהאות הראשונה של המחרוזת המקומית.

את האות הראשונה מקבלים מהשיטה **charAt()**, המייצגת אות במחרוזת על פי המיקום הנקבע בין הסוגריים. כלומר, charAt(0) שווה ערך לאות הראשונה במחרוזת.

דוגמה 17-04.html:

```
<html>
  <head>
   <script language="javascript">
     function text_mover( )
      {
        a = document.formi.texi.value ;
        a = a.substring(1,a.length) + a.charAt(0) ;
        document.formi.texi.value = a ;
      }
   </script>
   <title>
   </title>
  </head>
  <body>
    <form name="formi">
      <input type="text" name="texi" value="" />
      <input type="button" value="Roll text" onclick="text_mover()" />
    </form>
```

```
  </body>
</html>
```

בדוגמה זו, הלחצן בטופס קורא לפונקציה ()text_mover אשר מרכיבה מחרוזת חדשה על ידי שרשור שתי מחרוזות, וקובעת את המחרוזת החדשה כערך של שדה הטקסט בטופס. הפונקציה שבדוגמה **17-04.html** פועלת רק פעם אחת, ולכן יש ללחוץ על הלחצן עבור כל תנועה.

השיטה ()setInterval

השיטה **()setInetval** מאפשרת הפעלה של פונקציה במרווח זמן בפעולה מחזורית. לשיטה זו יש שני פרמטרים: הראשון הוא שם הפונקציה והשני הוא מירווח הזמן במילישניות עד להפעלה הבאה. מהרגע שהופעלה פונקציה באמצעות ()setInterval אין צוטרך בהפעלתה שוב. היא תופעל בצורה מחזורית במירווח הזמן שנקבע. ניתן גם להוסיף לחצן לעצירת הפעולה וזאת נעשה בעזרת השיטה ()clearInterval במידה והפונקציה ()setInterval מופעלת במשפט הצבה לתוך משתנה.

דוגמה 17-05.html:

```
<html>
  <head>
    <script language="javascript">
     function initTextMover() {
       x = setInterval('TextMover()',600) ;
      }

     function TextMover()
      {
       a = document.formi.texi.value ;
       a = a.substring(1,a.length) + a.charAt(0) ;
       document.formi.texi.value = a ;
      }
    </script>
    <title>
    </title>
  </head>
  <body>
    <form name="formi">
      <input type="text" name="texi" value="" />
      <input type="button" value="Roll text" onclick="initTextMover()" />
      <input type="button" value="Stop rolling" onclick="clearInterval(x)" />
    </form>
  </body>
</html>
```

עוד על אובייקט המחרוזת

השיטות **toUpperCase()** ו-**toLowerCase()** משמשות להסבת המחרוזת מאותיות רישיות (גדולות) לאותיות רגילות (קטנות) ולהיפך, לדוגמה:

```
misty = "aMeRica" ;
alert(misty.toLowerCase() + " " + misty.toUpperCase()) ;
```



תרשים 17.1

טיפול בטפסים

שדות טקסט

הגישה לשדה טקסט כאובייקט מתבצעת על ידי שימוש באובייקט document, נקודה, שם הטופס המכיל את השדה, נקודה ושם השדה.

אל השדה field אשר בטופס formi:

```
<form name="formi">
    <input type="text" name="field">
</form>
```

ניתן לגשת כ-document.formi.field, כדי לקבוע את ערך השדה כ-"hello". לשם כך נוסיף את השורה הזו:

```
document.formi.field.value="hello"
```

כלומר, ניתן לגשת אל המאפיין value של אובייקט השדה, לקבוע או לבדוק אותו. באותו אופן ניתן לפנות ל-Checkbox או ל-Textarea.

רשימה נגללת select

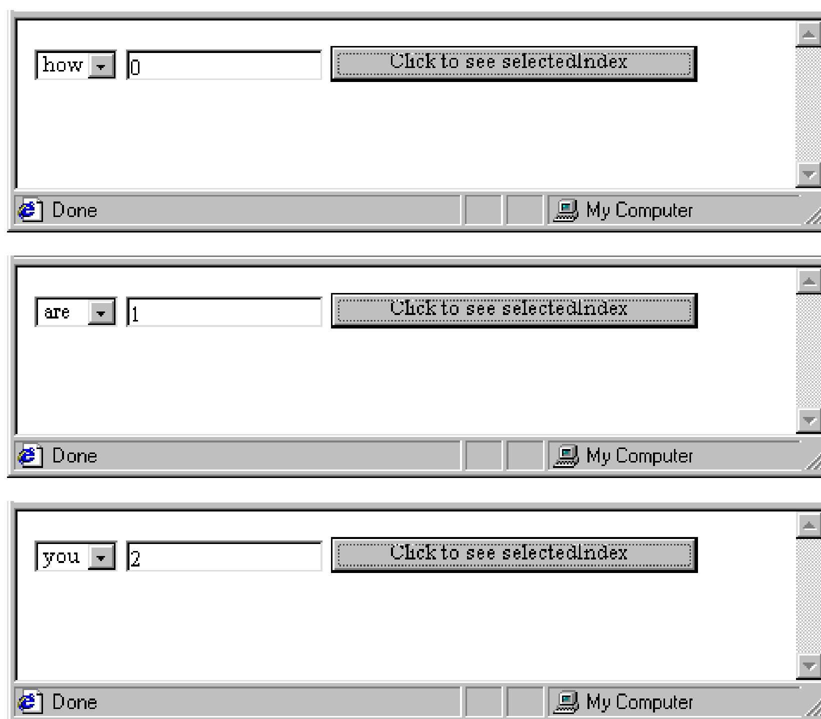
לרשימה נגללת יש לגשת באופן שונה מעט מאשר אל שדות טקסט. כדי לברר איזה ערך נבחר ברשימה, יש לגשת תחילה אל אובייקט הרשימה, בדומה לגישה אל אובייקט שדה הטקסט.

```
<form name="formi">
     <select name="droplist">
       <option value="to">how</option>
       <option>are</option>
       <option>you</option>
     </select>
</form>
```

אל השדה droplist אשר בטופס formi ניתן לגשת כ-document.formi.droplist, אולם כדי לברר את הערך הנבחר, יש להשתמש בשני מאפיינים. המאפיין הראשון הוא selectedIndex, המציין איזו מהאפשרויות (המסודרות **כמערך**) נבחרה. כלומר, אם נבחר באפשרות (האיבר) how ונציג את document.formi.droplist.selectedIndex נקבל 0, כיון שהאיבר הראשון במערך מסומן כמספר 0. הדוגמה הבאה ממחישה את השימוש בתכונה selectedIndex.

דוגמה 17-06.html:

```
<html>
  <head>
     <script language="javascript">
      function joe()
       {
        document.formi.field.value = document.formi.droplist.selectedIndex ;
       }
     </script>
    <title>
    </title>
  </head>
  <body>
    <form name="formi">
     <select name="droplist">
       <option value="to">how</option>
       <option>are</option>
       <option>you</option>
     </select>
     <input type="text" name="field" />
     <input type="button" onclick="joe()" value="Click to see selectedIndex" />
    </form>
  </body>
</html>
```



תרשים 17.2

כפי שניתן לראות בתמונות המסך, selectedIndex מחזירה את המספר הסידורי של הבחירה במערך האפשרויות המוצג.

כדי לקבל את הערך של האפשרות (האיבר) שנבחרה ולא את המספר הסידורי שלה, יש לפנות אל הרשימה כמערך:

```
document.formi.droplist.options[the number you want].value
```

המספר שיצוין בין הסוגריים המרובעים יפנה אל האפשרות המתאימה במערך select. אם רוצים לקבל את הערך של האפשרות שנבחרה על ידי המשתמש, יש לכתוב:

```
document.formi.droplist.options[document.formi.droplist.selectedIndex].value
```

המשפט document.formi.droplist.selectedIndex ייתן את המספר הסידורי של האפשרות הרצויה. הדוגמה הבאה מבהירה את דרך הפנייה לערך הרשימה.

דוגמה 17-07.html:

```
<html>
 <head>
    <script language="javascript">
     function joe()
      {
       var y ;
       var x ;
       y = document.formi.droplist.selectedIndex ;
       x = document.formi.droplist.options[y].value ;
       document.formi.texti.value = x ;
      }
    </script>
   <title>
   </title>
 </head>
 <body onload="joe()">
   <form name="formi">
    The chosen option is :
    <input type="text" name="texti" /><br />
    <select name="droplist" onchange="joe()">
      <option value="Welcome">Welcome</option>
      <option value="To">To</option>
      <option value="Jamaica">Jamaica</option>
      <option value="And" selected>And</option>
      <option value="Have">Have</option>
      <option value="A">A</option>
      <option value="Nice">Nice</option>
      <option value="Day">Day</option>
    </select>
   </form>
 </body>
</html>
```

לפניך תמונות המסך של הדוגמה. שים לב, כי הפעם נקבע ערך שדה הטקסט לפי ערך הבחירה ולא לפי מספרה הסידורי.



תרשים 17.3

שים לב לכך שהפונקציה ()joe הופעלה לראשונה במסגרת התגית <body>. הפעלה זו נועדה לטעון את השדה בטופס ברירת המחדל של שדה select. ללא הפעלה זו בעת הטעינה הראשונה של הדף, השדה היה נשאר ריק. נסה ותיווכח בעצמך.

לחצני רדיו - Radio buttons

הפנייה בתוכנית אל לחצני רדיו כדי ללמוד מי מהם נלחץ היא פעולה מסובכת יותר, כי אז יש לבדוק כל אחד מהם. לחצן שנלחץ מקבל ערך true במאפיין checked.

לחצני הרדיו, בדומה לפריטים ברשימת הגלילה, מסודרים כאיברים של מערך. עלינו להריץ לולאה על מערך הלחצנים הזה, מ- 0 עד לאיבר האחרון של המערך. בדוגמה הבאה, יש לולאה הפועלת על איברי המערך os המכילים את הלחצנים ובודקת את ערך התכונה checked של כל לחצן. רק בלחצן הנבחר הערך יהיה true.

ניצור את מערך הלחצנים os.

```
<form name="formi">
     Select an operating system<br />
     <input type="radio" name="os" value="Win95" onclick="joe()" />
      Win95 <br />
     <input type="radio" name="os" value="Win98" onclick="joe()" />
      Win98 <br />
     <input type="radio" name="os" value="WinNT" onclick="joe()" />
      WinNT <br /><br />
      You have selected
     <input type="text" name="texti" />
</form>
```

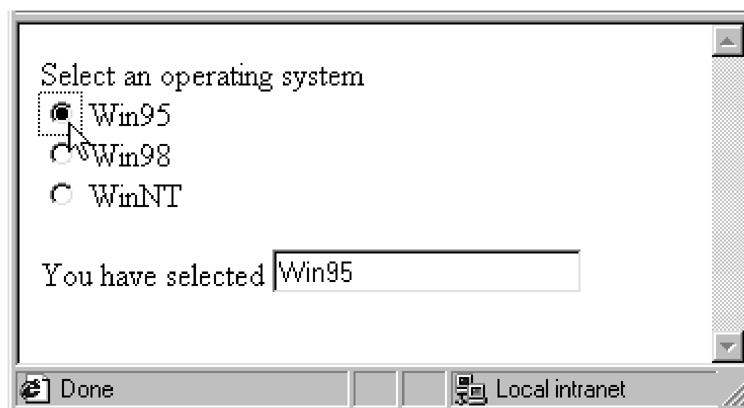
מכיון שהלחצנים מהווים מערך, הלחצן הראשון מסומן 0. לכן, אם נקבל בבדיקה את הערך 0, נדע שהמשתמש בחר ב-Win95 וכן הלאה. מספר האיברים במערך נמצא במאפיין length של os.

בדוגמה document.formi.os[I].value, מופיעה הלולאה המבצעת את הבדיקה ושולחת את ערך הלחצן הלחוץ אל שדה הטקסט texti:

```
for(i = 0 ;  i < document.formi.os.length ;  i++)
 {
   if(document.formi.os[i].checked)
     document.formi.texti.value = document.formi.os[i].value ;
 }
```

דוגמה 17-08.html:

```
<html>
 <head>
    <script language="javascript">
     function joe()
      {
        for(i = 0 ;  i < document.formi.os.length ;  i++)
         {
           if(document.formi.os[i].checked)
             document.formi.texti.value = document.formi.os[i].value ;
         }
      }
    </script>
  <title>
  </title>
 </head>
 <body>
  <form name="formi">
    Select an operating system<br />
    <input type="radio" name="os" value="Win95" onclick="joe()" />
     Win95 <br />
    <input type="radio" name="os" value="Win98" onclick="joe()" />
     Win98 <br />
    <input type="radio" name="os" value="WinNT" onclick="joe()" />
     WinNT <br /><br />
     You have selected
    <input type="text" name="texti" />
  </form>
 </body>
</html>
```



תרשים 17.4

פרק 18

אובייקט התאריך

לעיתים קרובות אנו זקוקים למידע זמן ביישומים שונים, כמו שעה או תאריך. JavaScript מכילה אובייקט מיוחד המספק את כל המידע הקשור בזמן, זהו **אובייקט התאריך** (Date object). את אובייקט התאריך ניתן לייצר בדומה לאובייקט התמונה על ידי שימוש במילה השמורה new.

```
a = new Date() ;
```

אם נבקש להציג את האובייקט נקבל מחרוזת המפרטת את הזמן המדויק כולל שניות, דקות, שעות ותאריך, לדוגמה:

```
Mon Feb 12 21:14:37 UTC+0200 2001
```

הדוגמה הבאה תציג בראש מסמך את מחרוזת התאריך.

דוגמה 18-01.html:

```
<html>
  <head>
    <script language="javascript">
      a = new Date() ;
      document.write(a) ;
    </script>
    <title>
    </title>
  </head>
</html>
```

כך תיראה התוצאה בדפדפן:



תרשים 18.1

ברגע שנוצר אובייקט התאריך, ניתן להשתמש במספר שיטות חשובות המציגות את הגורמים השונים המרכיבים את האובייקט. אם נרצה לברר את השנה הנוכחית, נשתמש בשיטה **getYear()**. כלומר: document.write(a.getYear()).

שיטות נוספות של האובייקט **Date()**:

השיטה	טווח לרבים
getMonth()	0..11
getDate()	1..31
getHours()	0..23
getMinutes()	0..59
getSeconds()	0..59

הערה

בשיטה **getMonth()** ספירת החודשים מתחילה ב- 0.

שעון

כדי ליצור שעון יש לבחור את דרך הצגתו. כלומר, אם נבחר בשיטה document.write(), לא נוכל לעדכן את השעון אלא להציגו באופן חד-פעמי, כיון שהשיטה write() רק מחוללת מסמך HTML ואינה מסוגלת לעדכן אותו. לכן, ניתן להציג את השעון בצורה דינמית רק בתוך שדה טקסט, שכזכור ניתן לעדכן אותו בצורה דינמית על ידי קביעת ערך המאפיין value שלו, או על ידי שורת הסטטוס שערכה נקבע על ידי window.status.

שעונים רבים ברשת משתמשים בתמונות המשתנות דינמית על פי שעון המחשב. קוד כזה מחייב בדיקה חוזרת של מחרוזת השעה והתאמת התמונות במקום הנכון. ניישם שעון ב**שורת הסטטוס** (status bar), אך ניתן כמובן להסב את הקוד להצגת השעון בכל דרך אחרת.

בכל ריצה של הפונקציה יש לייצר את אובייקט התאריך, מכיון שאובייקט התאריך הוא למעשה מחרוזת המייצגת את הזמן המדויק ברגע יצירתו בלבד.

```
function shaon()
{
  tarich = new Date() ;
```

לאחר מכן, ניצור מחרוזת שתכיל את השעות, הדקות והשניות מתוך אובייקט התאריך.

```
a = tarich.getHours() + ":" + tarich.getMinutes() + ":" + tarich.getSeconds() ;
```

לבסוף, כל מה שנותר הוא לקבוע את שורת הסטטוס כמשתנה המחרוזת a ולקרוא שוב לפונקציה על ידי השיטה **setInterval()**.

```
    window.status = a ;
    setInterval('shaon()',1000) ;
   }
```

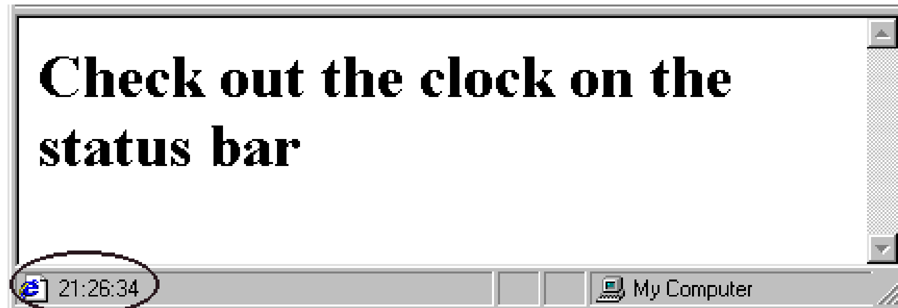
לפניך הקוד המלא ליצירת שעון בשורת הסטטוס. המספר 1000 בפונקציה setInterval() מפעיל את הפונקציה shaon() כל 1000 מילישניות. הקריאה לפונקציה המבצעת נמצאת בתגית <body> במתכונת של ההדק onload.

דוגמה 18-02.html:

```
<html>
  <head>
    <script language="javascript">
      function initShaon() {
         setInterval('shaon()',1000) ;
       }

      function shaon()
       {
         tarich = new Date() ;
         a = tarich.getHours() + ":" + tarich.getMinutes() + ":" + tarich.getSeconds() ;
         window.status = a ;
       }
    </script>
    <title>
    </title>
  </head>
  <body onload="initShaon()">
   <h1>Check out the clock on the status bar</h1>
  </body>
</html>
```

שים לב לשורת הסטטוס בתרשים המסך:



תרשים 18.2

כדאי שוב לשים לב להפעלת השיטה ()setInterval המופעלת מתוך הפונקציה ()initShaon ומפעילה את הפונקציה ()shaon. לחילופין, ניתן היה לכתוב כך את הפעלת הפונקציה ()shaon:

```
<body onload=" setInterval('shaon()',1000)">
```

אם הבחנת, השעון שבדוגמה מראה תאריך במבנה hh:mm:ss, אבל מה קורה כאשר הערך להצגה הוא עם ספרה אחת? איך תוצג השעה שמונה בבוקר, 5 דקות ו-7 שניות? כך: 08:05:07 או 8:5:7?

בקובץ **18-03.html** תמצא קוד משופר המציג את השעון בשורת הסטטוס:

```
function shaon()
  {
    tarich = new Date() ;
    hh = tarich.getHours() ;
    mm = tarich.getMinutes() ;
    ss = tarich.getSeconds() ;
    if (hh < 10) hh = "0" + hh ;
    if (mm < 10) mm = "0" + mm ;
    if (ss < 10) ss = "0" + ss ;
    a = hh + ":" + mm + ":" + ss ;
    window.status = a ;
}
```

עוד על אובייקט התאריך

בנושא התאריך נציג שתי שיטות נוספות: **getTime()** ו-**setTime()**.

השיטה **getTime()** מחזירה מחרוזת המייצגת את מספר אלפיות השנייה שחלפו מאז 1.1.1970, לדוגמה: 886774180650.

השיטה **setTime()** קובעת את התאריך לפי הנוסחה: 1.1.1970 ועוד מספר אלפיות השנייה המוזנות בין הסוגריים של השיטה, אשר מתורגמות לימים.

כדי לקבל את התאריך הנמוך ביותר, נכתוב:

```
tarich = new Date() ;
tarich.setTime(0) ;
```

והערך שיתקבל במשתנה tarich יהיה:

```
Thu Jan 01 02:00:00 UTC+0200 1970
```

בדוגמה הבאה נייצר תאריך המייצג חודש אחד לאחר התאריך הנוכחי. לשם כך, נייצר אובייקט תאריך המכיל את התאריך הנוכחי ונהפוך אותו למחרוזת של אלפיות שנייה על ידי שימוש בשיטה **getTime()**.

```
a = new Date() ;
a_milisecond_string = a.getTime() ;
```

לאחר מכן, נייצר מחרוזת המייצגת אלפיות שנייה בחודש. נעשה זאת על ידי כפל 1000 אלפיות שנייה (כלומר שנייה) ב-60 כדי להגיע לדקה אחת, נכפול ב-60 כדי להגיע לשעה, נכפול ב-24 כדי להגיע לימָמה ונכפול ב-31 כדי להגיע לחודש ימים. את מחרוזת זו נוסיף למחרוזת המייצגת את התאריך הנוכחי a.getTime().

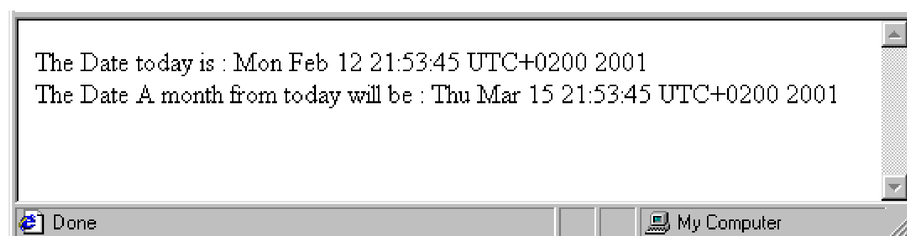
לבסוף, נקבע את ערך אובייקט התאריך למחרוזת החדשה שיצרנו, אשר מייצגת כעת את מספר אלפיות השנייה שחלפו מ- 1.1.1970 ועד חודש קדימה מהתאריך הנוכחי. כעת נותר רק להציג את התאריך.

דוגמה 18-04.html:

```
<html>
  <head>
    <script language="javascript">
      a = new Date() ;
      document.write("The Date today is : " + a) ;
      a.setTime(a.getTime() + (1000*60*60*24*31)) ;
      document.write("<br />")
      document.write("The Date A month from today will be : " + a) ;
    </script>
    <title>
    </title>
  </head>
</html>
```

השימוש בשתי שיטות אלו יבוא לידי ביטוי בחישוב של תאריכים מרוחקים לעתיד או בעבר.

התוצאה בדפדפן תוצג כך:



תרשים 18.3

פרק 19

מסגרות - בניית יישום בצד הלקוח

טכנולוגיית **המסגרות** (frames) מאפשרת הצגת מספר דפי HTML על מסך אחד. למרות שהכרת טכנולוגיה זו היא חלק מדרישות הקדם של הספר, נחזור בקצרה על הנושא. מי שאינו מכיר את טכנולוגיית המסגרות, רצוי שיחזור ויעיין בפרק 10 שבספר זה.

<frameset> - הפקודה פותחת קבוצת מסגרות. לפקודה זו מספר מאפיינים, כגון rows ו-cols הקובעים אם החלוקה היא לשורות או לטורים, או המאפיין frameborder המאפשר ביטול הגבולות בין המסגרות.

<frame /> - הפקודה מגדירה את תכונות המסגרת עצמה וכוללת מאפיינים, כגון src, הקובעים איזה מסמך יופיע במסגרת. המאפיין name, למשל, מאפשר לקבוע שם למסגרת, כדי שאפשר יהיה להתייחס אליה בזמן ריצה.

דוגמה:

```
<frameset rows="20%,*">
   <frame name="yaron" src="kooki.html" />
   <frame name="moshe" src="joe.html" />
</frameset>
```

הפעלת קישורית בין מסגרות מתבצעת על ידי שימוש במאפיין **target** של התגית **<a>**. דוגמה לקישור למסגרת בשם moshe ניתן לראות בקטע הקוד הבא.

```
<a href="http://www.hod-ami.co.il" target="moshe">Click for yahoo</a>
```

שמירת נתונים במסגרות סטטיות

אחד השימושים העיקריים בטכנולוגיית המסגרות הוא מסגרת תפריט קבועה המאפשרת גלישה במסגרות דינמיות. ניתן להשתמש במסגרות הסטטיות, כדי לשמור נתונים לאורך כל זמן השימוש באתר. התגית **<frameset>** מגדירה אובייקט בשם parent. כל התייחסות למסגרת שהוגדרה בעזרת התגית **<frame />** תחת התגית **<frameset>** תהיה דרך אובייקט **parent**. מכיון שכל מסגרת היא חלון, כלומר היא אובייקט **document**, ההתייחסות לתוכן המסגרת תהיה:

```
parent.frame_name.document
```

ניתן גם להשתמש במילה **top**, כאשר עוסקים במסגרות מקוננות ורוצים להגיע לקבוצת המסגרות העליונה.

בדוגמה הבאה ניצור מערכת מסגרות המחלקת את המסך לשתי עמודות. למסגרת השמאלית נקרא menubar והיא תכיל את המסמך **menu.html**. למסגרת הימנית נקרא mainscreen והיא תכיל את המסמך **main.html**.

דוגמה 19-01.html: תבנית המסגרות

```
<html>
  <head>
    <title>
    </title>
  </head>
  <frameset cols="50%,*">
    <frame name="menubar" src="menu.html" />
    <frame name="mainscreen" src="main.html" />
  </frameset>
</html>
```

בדוגמה הבאה ניצור את מסמך **menu.html**. מסמך זה יכיל את הקישורים לשני המסמכים (**main.html** ו-**second.html**) תוך הפניה למסגרת הימנית target="mainscreen".

במסמך ישנו גם **טופס** (form) בשם information, אשר יאפשר הצגת שני מונים: maincounter ו-secondcounter. מונים אלה יכילו את מספר הפעמים שהופעלו הקישורים בדף menu.html כדי לראות את כל אחד משני הדפים שבאתר. בשדה נוסף בשם current, יוצג שם הדף הנוכחי במסגרת הימנית הדינמית.

בתוך חלק הקוד בקובץ **menu.html** ניתן לראות כי יצרנו שני משתנים (main_counter_variable, second_counter_variable), אשר יכילו את מספר הפעמים שהמבקר נמצא בדפים המשניים. הפונקציה ()counter_display (גם היא בדף **menu.html**) מציגה את ערכי המשתנים בשדות הטקסט אשר בדף, אולם אין במסמך זה אף קריאה לפונקציה. הקריאה לפונקציה תתבצע מהדפים המשניים בזמן שנטען אותם.

דוגמה menu.html: מסמך התפריט - ימוקם במסגרת הסטטית menu.

```
<html>
 <head>
    <script language="javascript">
     var main_counter_variable = 0 ;
     var second_counter_variable = 0 ;
      function counter_display()
       {
        document.information.maincounter.value = main_counter_variable ;
        document.information.secondcounter.value = second_counter_variable ;
       }
    </script>
   <title>
   </title>
 </head>
 <body bgcolor="black" text="white" link="white" vlink="white">
  <font size="5">
   MENU
  </font>
   <hr />
   <a href="main.html" target="mainscreen">main page</a>
   <br />
   <a href="second.html" target="mainscreen">second page</a>
   <hr />
   <form name="information">
     Number of visits to main page :
    <input type="text" name="maincounter" size="3" value="1" />
    <br />
     Number of visits to second page :
    <input type="text" name="secondcounter" size="3" value="0" />
    <br />
     Current page :
    <input type="text" name="current" />
   </form>
 </body>
</html>
```

בדוגמה הבאה ניצור את המסמך שיופיע ראשון במסגרת הימנית הדינמית, ובדוגמה שאחריה ניצור את המסמך הנוסף המשתתף באתר. בשני המסמכים נעדכן את משתני המונים הנמצאים במסמך שבמסגרת הסטטית menu. הפנייה למשתנה שנמצא במסגרת אחרת מתבצעת בעזרת המילה parent, המציינת כי יש לפנות להורה המסגרת שנמצא ברמה אחת גבוהה יותר. לאחר שעלינו רמה אחת ניתן לפנות למסגרת הרלוונטית menu ולמשתנה שבה, main_counter_variable.

התחביר המלא הוא:

```
parent.menubar.main_counter_variable += 1 ;
```

כעת, נקבע את הערך המתאים לשדה current במסגרת menu על ידי המשפט :

```
parent.menubar.document.information.current.value = "Home Page" ;
```

נקרא לפונקציה ()menu.counter_display על ידי שימוש באותו תחביר.

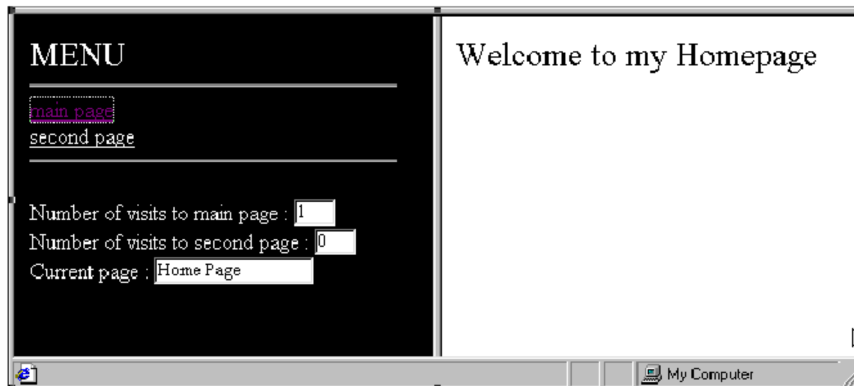
דוגמה main.html. מסמך דף הבית ימוקם במסגרת הדינמית mainscreen :

```
<html>
 <head>
     <script language="javascript">
      parent.menubar.main_counter_variable += 1 ;
      parent.menubar.document.information.current.value = "Home Page" ;
      parent.menubar.counter_display() ;
     </script>
   <title>
   </title>
 </head>
 <body bgcolor="white">
   <font size="5">
    Welcome to my Homepage
   </font>
 </body>
</html>
```

דוגמה second.html. בקריאת העמוד הנוסף באתר יופיע במסגרת הדינמית mainscreen :

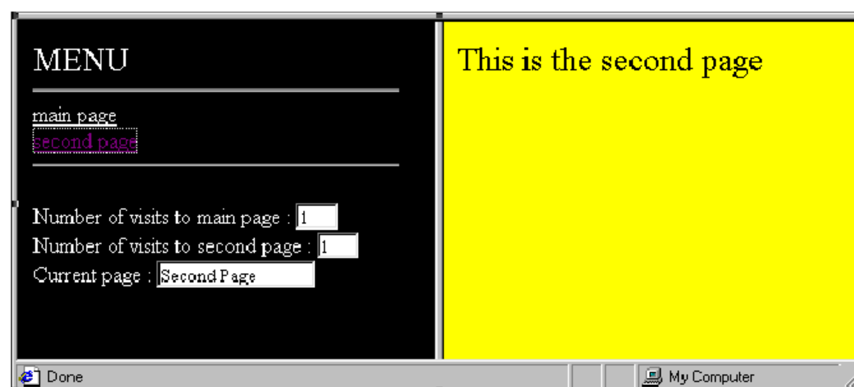
```
<html>
 <head>
     <script language="javascript">
      parent.menubar.second_counter_variable += 1 ;
      parent.menubar.document.information.current.value = "Second Page" ;
      parent.menubar.counter_display() ;
     </script>
   <title>
   </title>
 </head>
 <body bgcolor="yellow">
 <font size="5">
  This is the second page
 </font>
 </body>
</html>
```

לפניך התוצאה בדפדפן:



תרשים 19.1

וזו התוצאה לאחר לחיצה על קישורית ל-second page.



תרשים 19.2

כמה טיפים על מסגרות

לצד היתרונות של עבודה במסגרות יש לציין גם את החסרונות. הסיבה הבולטת היא בעיות בשמירת רשימות **מועדפים** (Favorites), בעיות בהדפסה ואיטיות מסוימת בטעינת הדפים הראשונים. למרות זאת, טכנולוגיה זו נוחה מאוד ומאפשרת שמירת נתונים בצד מחשב הלקוח.

בעיה נוספת עלולה להתעורר עקב הודעות שגיאה הנובעות מבדיקת משתנים או ערכים, הנמצאים במסגרות שונות שטרם נטענו אל הדפדפן. הודעות שגיאה אלו לרוב אינן מפריעות לריצת הקוד, אך אינן מקצועיות.

עוד על עבודה במסגרות תוכל לקרוא בספר **עיצוב ממשק באינטרנט** בהוצאת **הוד-עמי**.

פרק 20

חלונות

JavaScript מאפשרת לפתוח חלונות נוספים של הדפדפן ובכך להעשיר את הממשק למשתמש. חשוב לזכור כי חלונות דפדפן נוספים אינם תמיד חביבים על המשתמש, וניהול לא נכון שלהם עלול לגרום לבלבול. קרא על כך בספר **עיצוב ממשק באינטרנט** בהוצאת **הוד-עמי**.

למשל, חלון חדש אפשר לפתוח בצורה הבאה:

```
<a href="butter.html" target="_blank">open a new windows</a>
```

Javascript מציעה מעבר לפתיחת חלון גם יכולת להתייחס אליו, אל האובייקטים שבו.

בדוגמה הבאה נגדיר את המשתנה x כאובייקט חלון חדש:

```
x = window.open("test.html","testwindow","width=200, height=200") ;
```

אם נרשום שורת קוד זו במסמך HTML, ייפתח חלון חדש.

האובייקט x מייצג את חלון בשם testwindow המכיל את המסמך test.html. התחביר הוא:

```
object = window.open("content","window name","window attributes") ;
```

כפי שבוודאי שמת לב, החלון שנוצר הוא חלון בסיסי שאינו מכיל לחצנים או פסי גלילה. ניתן להעניק לחלון חדש מאפיינים רבים באמצעות פרמטרים הנכתבים בסוגריים ובין גרשיים, לאחר פקודת הפתיחה של החלון. ניתן לראות זאת בדוגמה שלאחר הטבלה.

להלן רשימת המאפיינים והפרמטרים האפשריים. שים לב שבמקום yes או no, אפשר לכתוב 1 או 0 בהתאמה (ברירת המחדל מסומנת בהדגשה):

רוחב החלון בפיקסלים	width=
גובה החלון בפיקסלים	height=
אפשרות שינוי גודל	resizable= (yes/**no** – 1/**0**)
פסי גלילה	scrollbars= (yes/**no** – 1/**0**)
שורת תפריטים	menubar= (yes/**no** – 1/**0**)
שורת לחצנים	toolbar= (yes/**no** – 1/**0**)
שורת מצב	status= (yes/**no** – 1/**0**)
שורת מיקום	location= (yes/**no** – 1/**0**)
מרחק החלון משמאל המסך בפיקסלים בדפדפנים מדור 4 ומעלה	top=
מרחק השדה מראש המסך בפיקסלים בדפדפנים מדור 4 ומעלה	left=

דוגמה:

```
x = window.open("1.html","joe","width=100,height=100,resizable=1,toolbar=1") ;
```

אם לא יצוין ערך לפרמטר (yes או no ובהתאמה: 1 או 0), ייקבע ערך ברירת המחדל no או 0.

אפשר גם לפתוח חלון ריק בצורה הבאה:

```
x = window.open() ;
```

דרך אחרת לפתוח חלון היא:

```
<a href="javascript:void(0)" onclick="window.open();">bla bla bla</a>
```

משפט זה יגרום לפתיחת חלון ריק. כדי להתייחס אליו יש צורך לתת לו שם בצורה הבאה:

```
<a href="javascript:void(0)" onclick="x=window.open();">bla bla bla</a>
```

משפט זה לא רק שייפתח חלון ריק אלא גם ייתן לו שם – x שבעזרתו נוכל להתייחס לאותו חלון.

קובץ **20-01.html** מרכז מספר אפשרויות לפתיחת קובץ :

```
<html>
  <head>
      <script language="javascript">
        function NewAndEmptyNoName()
          {
            window.open() ;
         }

        function NewWithName(what2open)
          {
            mySon = window.open(what2open) ;
         }

      </script>
          <title>
    </title>
  </head>
<body>
<p>Open New Window:</p>
<a href="javascript:void(0)" onclick="window.open('amit.html')">
   blank window with no name</a><br />
<a href="javascript:void(0)" onclick="newWin=window.open('amit.html')">
   blank window with a name</a><br />
<a href="javascript:void(0)" onclick="NewAndEmptyNoName()">
   a new and empty window with a function</a><br />
<a href="javascript:void(0)" onclick="NewWithName('amit.html')">
   a new window with a function</a><br />
</body>
</html>
```

הפעל את הדף ואת הקישורים שבו וראה מה שמתרחש. הפונקציה **NewWithName()** פותחת חלון חדש עם תוכן הקובץ ששמו הועבר אליה כפרמטר. מבחינה ויזואלית הקישור הראשון, השני והרביעי מציגים את אותו דבר. אבל, הקישור השני והרביעי נותנים שם לחלון החדש, דבר שמבטיח שנוכל להתייחס אליו בהמשך.

העברת מידע בין חלונות

בדומה לעבודה עם **מסגרות** (frames), ניתן לפנות לפונקציות ולערכי משתנים בחלונות שונים תוך שימוש בשם משתנה החלון. כאשר נפתח חלון: ...)x=window.open, נוכל לפנות אל פונקציות שנמצאות בו על ידי פקודה במבנה הבא:

```
x.function_name
```

נוכל גם לשנות מאפיינים, כגון צבע רקע, על ידי פקודה בתחביר הזה:

```
x.document.bgColor = 'red' ;
```

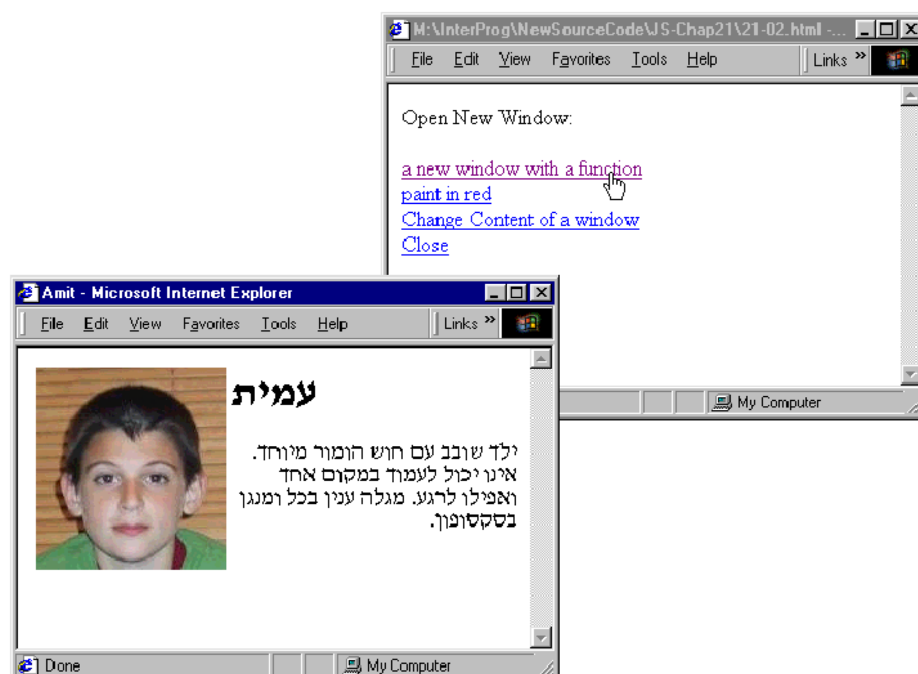
קובץ **20-02.html** מדגים פתיחת חלון, שינוי מאפיינים והפעלת שיטות מהחלון שפתח אותו.

```
<html>
  <head>
      <script language="javascript">
        function NewWithName(what2open)
          {
            mySon = window.open(what2open) ;
         }

        function paintRed()
          {
            mySon.document.bgColor = "red" ;
         }

        function changeContent()
          {
            mySon.document.location = "eyal.html" ;
          }
      </script>
          <title>
    </title>
  </head>
<body>
<p>Open New Window:</p>
<a href="javascript:void(0)" onclick="NewWithName('amit.html')">a new window with a
function</a><br />
<ul>
<li><a href="javascript:void(0)" onclick="paintRed()">paint in red</a><br /></li>
<li><a href="javascript:void(0)" onclick="changeContent()">Change Content of a
window</a><br /></li>
<li><a href="javascript:void(0)" onclick="mySon.close()">Close</a><br /></li>
</ul>
</body>
</html>
```

כדי לראות איך באמת זה עובד, יהיה עליך להפעיל את הקובץ **20-02.html** ולהפעיל את הקישור הראשון שבו. עכשיו סדר את שני החלונות על המסך, כך שלא יהיו אחד על השני. לחיצה על אחד הקישורים ברשימת התבליטים תשפיע על מראה החלון שנפתח. קישורים אלה יעבדו רק אם ייפתח חלון חדש על ידי הפעלת הקישור העליון.



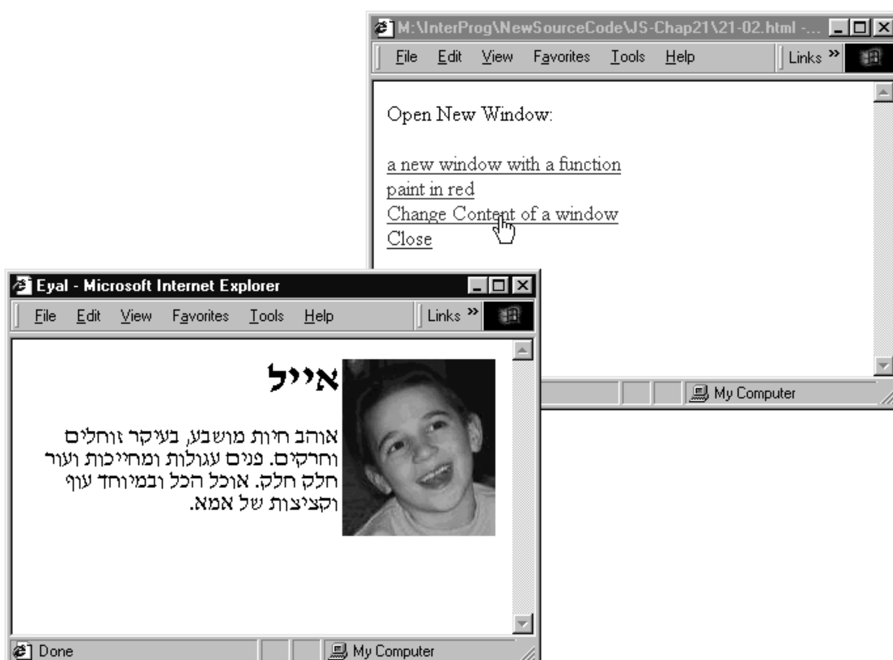
תרשים 20.1

עכשיו הפעל את הקישור השני.



תרשים 20.2

לחץ על הקישור השלישי



תרשים 20.3

תוכנית **20-03.html** מדגימה עבודה עם שני חלונות תוך מתן דגש לנקודות הבאות:

1. מתן גישה לחלון רק אם הוא קיים. בעבודה עם חלונות, יכול להיות מצב בו נרצה לגשת לחלון שנפתח ונסגר. כמובן שניסיון שכזה יציג שגיאה. הפתרון: שימוש במאפיין closed של האובייקט window כדי לדעת אם החלון נסגר.

```
        if (newWin.closed)
            alert("Sorry") ;
          else
            newWin.focus() ;
```

2. חסימת האפשרות לפתוח חלון נוסף יותר מפעם אחת. לצורך זה נבנה מנגנון המבוסס על משתנה בוליאני בשם boolNeed2Open.

```
<script language="javascript">
        var boolNeed2Open ;
        boolNeed2Open = true;

        function makeNewWindow()
         {
           if (boolNeed2Open)
            {
              newWin = window.open() ;
               statements
              boolNeed2Open = false ;
            }
         }
```

שים לב שהמשתנה boolNeed2Open מוגדר מחוץ לפונקציה כדי שיהיה משתנה גלובלי. בתחילת הדרך ערכו true, והמשמעות היא שברגע הפעלת הפונקציה makeNewWindow() התנאי מתקיים ונפתח חלון חדש. לאחר מכן, מוצב במשתנה boolNeed2Open הערך false ובכך נמנע מהפונקציה makeNewWindow() לפתוח יותר מחלון אחד.

3. בעיה נוספת עשויה להתעורר, כאשר המשתמש יפעיל פונקציה המנסה לגשת לחלון שלא נפתח כלל. בסעיף 1 מצאנו פתרון לניסיון לגשת לחלון שנפתח וגם נסגר. עכשיו, ננסה למנוע מהמשתמש הפעלת פונקציה לחלון שכלל לא נפתח. הפתרון: הסתרת הקישור להפעלת הפונקציה הניגשת לחלון.

```
<html>
  <head>
      <script language="javascript">
        var boolNeed2Open ;
        boolNeed2Open = true;
        function makeNewWindow()
```

```
        {
          if (boolNeed2Open)
           {
             newWin = window.open() ;
              statements
             inFocus.style.visibility="Visible" ;
             boolNeed2Open = false ;
           }
        }

</script>
     <title>
     </title>
   </head>
<body>
<a href="javascript:void(0)" onclick="makeNewWindow()">New Window</a>
<br /><br />
<div id="inFocus" style="visibility:hidden">
    <input type="button" value="The new window in focus"
        onclick="newWinInfocus()" />
    <br /><br />
    <input type="button" value="Close new window"
       onclick="newWin.close(); boolNeed2Open=true" />
    <br /><br />
</div>
<input type="button" value="Close the current window" onclick="self.close()" />
<br /><br />
</body>
</html>
```

הלחצן שתחת התגית <div> מוסתר באמצעות המאפיין visibility. עם פתיחת החלון החדש על ידי הפונקציה ()makeNewWindow "מתגלים" הלחצנים הקשורים בהפעלתו. אז המשתמש יכול להפעיל פונקציה הניגשת לאותו חלון.

התוכנית הלא גדולה הזאת עושה דבר קטן מאוד: היא מעבירה את המיקוד מהחלון הראשי לחלון החדש שנפתח, אך ראה כמה "לוגיסטיקה" יש מסביב כדי למנוע שגיאות בעת הרצה. להלן התוכנית המלאה הנמצאת בקובץ **20-03.html**:

```
<html>
  <head>
      <script language="javascript">
        var boolNeed2Open ;
        boolNeed2Open = true;
        function makeNewWindow()
         {
          if (boolNeed2Open)
```

```
        {
          newWin = window.open() ;
          newWin.document.writeln("<html>") ;
          newWin.document.writeln("<head>") ;
          newWin.document.writeln("<title>The Second HTML page</title>") ;
          newWin.document.writeln("</head>") ;
          newWin.document.writeln("<body>") ;
          newWin.document.writeln("<h1>U c what I c?</h1>") ;
          newWin.document.writeln("</body>") ;
          newWin.document.writeln("</html>") ;
          inFocus.style.visibility="Visible" ;
          boolNeed2Open = false ;
        }
      }

     function newWinInfocus()
      {
        if (newWin.closed)
            alert("Sorry") ;
          else
            newWin.focus() ;
      }
    </script>
    <title>
    </title>
  </head>
<body>
<a href="javascript:void(0)" onclick="makeNewWindow()">New Window</a>
<br /><br />
<div id="inFocus" style="visibility:hidden">
   <input type="button" value="The new window in focus"
     onclick="newWinInfocus()" />
   <br /><br />
   <input type="button" value="Close new window"
    onclick="newWin.close(); boolNeed2Open=true" />
   <br /><br />
</div>
<input type="button" value="Close the current window" onclick="self.close()" />
<br /><br />
</body>
</html>
```

חומר לימוד נוסף בנושא **HTML** בהוצאת **הוד-עמי**:

JavaScript למפתחי אתרים באינטרנט

JavaScript המדריך השלם

חלק 4 - ASP

פרק 21

הכרת טכנולוגיית ASP

כדי להבין את משמעות טכנולוגיית ASP, יש צורך להבין את פרוטוקול HTTP. פרוטוקול HTTP הוא אחד הפרוטוקולים הפשוטים ביותר ובעיקרון פועל כשרת קבצים. כלומר, ניתן להקביל את פרוטוקול HTTP לשרת המאחסן קבצים והיכול לשלוח קובץ עבור כל בקשה המגיעה מלקוח.

נתאר תהליך פשוט של גלישה ברשת לאתר הבית של YAHOO.

בשלב הראשון, מקליד הגולש את הכתובת http://www.yahoo.com בשדה הכתובת בדפדפן.

מתחת לפני השטח, פונה הדפדפן לשרת ה-HTTP של YAHOO. הפנייה היא בקשת Get. כלומר, הדפדפן מבקש לקבל דבר מה מהשרת. תחביר הפרוטוקול המדויק נראה כך :

Get / HTTP/1.1



תרשים 21.1

בקשת HTTP בנויה מהמילה Get המייצגת את בקשת הלקוח לקבל קובץ, מ**לוכסן** (Slash) ומגרסת הפרוטוקול (http/1.1). מכיון שבקשה זו מהשרת היא בקשה כללית ולא בקשה לקבל קובץ מסוים, על השרת להחליט איזה קובץ לשלוח ללקוח. חשוב להבין כי שרת הוא מחשב ככל מחשב, ולעיתים אינו חזק או מתוחכם יותר מהמחשב הנמצא בכל בית. האלמנט המבדיל בין מחשב רגיל לבין שרת הוא תוכנה המותקנת על המחשב ומאפשרת לאותו מחשב לקבל בקשות מלקוחות ולשלוח קבצים כתשובה. יש הרבה מאוד תוכנות שרת וחלקן אף מתאימות למחשבי PC בעלי מערכת הפעלה Windows 95/98.

בכל תוכנת שרת יש אפשרות לקבוע איזה קובץ יישלח כתשובה לבקשת Get כללית. קובץ זה מוגדר ברוב תוכנות השרת כ-Default Document או כ-Index וברוב המקרים הוא קובץ HTML.

בשלב הבא, מחזיר השרת את הקובץ המוגדר כ-Default Document אל הלקוח. הקובץ עובר בפורמט **ASCII** (American Standard Code for Information Interchange). פורמט ASCII מגדיר ערכים מספריים המייצגים כל אות, כך שלמעשה נשלחות ללקוח קבוצות מספרים אשר הוא מתרגם חזרה לאותיות, ומרכיב מהן קובץ טקסטואלי. קובץ טקסטואלי זה מכיל את ההוראות לדפדפן, כלומר HTML. עם קבלת הקובץ, ממלא הדפדפן אחר ההוראות ומציג בפני הגולש את האתר.



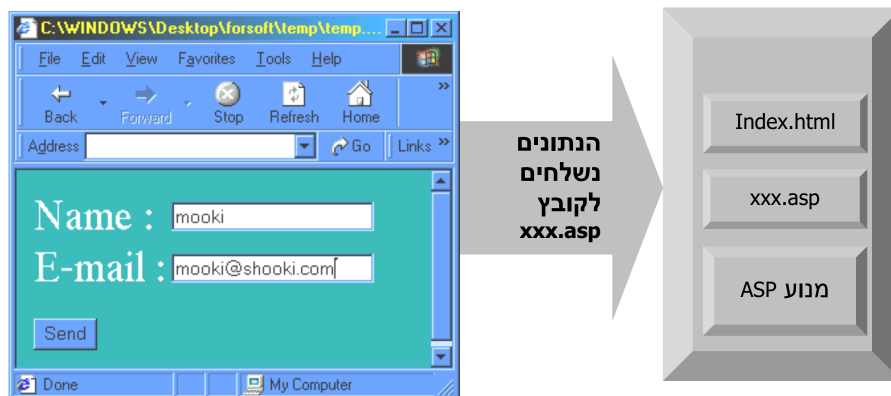
תרשים 21.2

בטכנולוגיה בסיסית זו, יש צורך לכתוב מספר רב של מסמכי HTML. ישנם חסרונות רבים בכתיבת HTML, או כפי שמקובל לכנות טכנולוגיה זו בעגה המקצועית - **אתרים סטטיים**. החסרונות נעים מהצורך לעדכן ידנית נתונים משתנים כגון תאריך, ועד לבעיות חמורות יותר, כגון חוסר היכולת להעביר נתונים מהלקוח לשרת ולחולל דפים שהתוכן שלהם משתנה על פי תנאים שונים.

טכנולוגיית ASP מאפשרת לבצע דברים רבים. העיקרון מאחורי ASP הוא לבנות קבצי HTML בזמן הבקשה מהלקוח (On-The-Fly או On-Demand). כלומר, **אין קבצי HTML מוכנים אלא קבצי HTML נוצרים עבור כל בקשת לקוח**. תפישה זו מאפשרת לבנות יישומים נרחבים על סביבת Web, כולל שימוש במסדי נתונים, קבלת נתונים מלקוחות, קישור למערכות שונות, שליחת דואר ועוד.

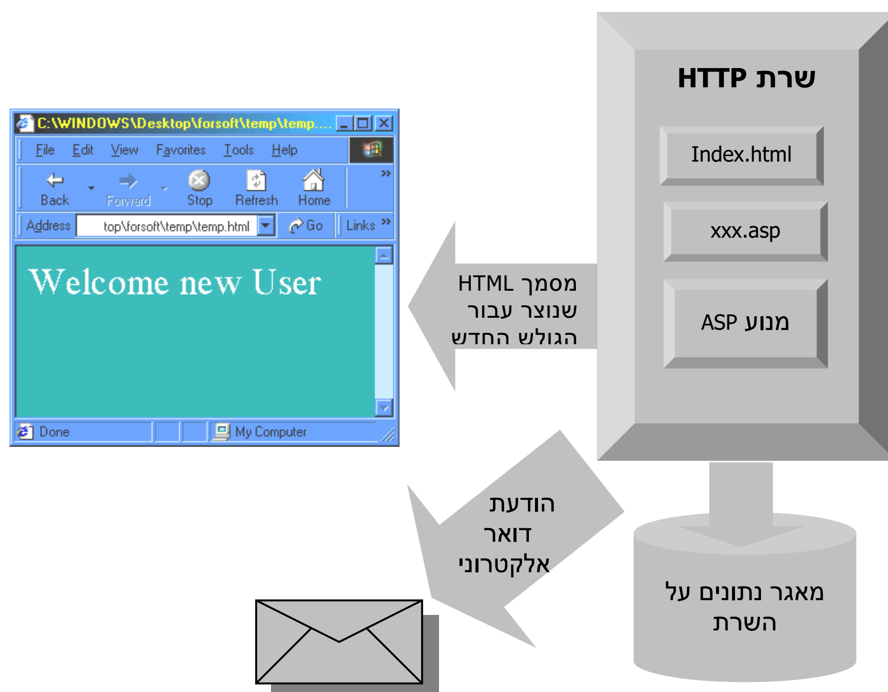
דוגמה טובה לאתר המשתמש ביכולות ASP, היא אתר בו המשתמש נרשם באתר באמצעות מילוי טופס. לאחר ההרשמה נשלחת הודעת דואר אלקטרוני אל הלקוח ומברכת אותו על הצטרפותו.

בשלב הראשון, הלקוח שולח בקשת Get רגילה ומקבל מסמך HTML רגיל ובו טופס הזדהות/הרשמה. כאשר הלקוח ממלא את הטופס ולוחץ על לחצן **שלח טופס**, נשלחים הנתונים אל קובץ בשרת. לקובץ זה סיומת ASP הגורמת למנוע ASP להתעורר ולמלא את ההוראות הרשומות בקובץ. פעולה זו מתבצעת על השרת.



תרשים 21.3

בשלב הבא, בודק מנוע ASP על פי ההוראה הרשומות בקובץ **xxx.asp**, אם שם המשתמש שהקליד הגולש קיים במאגר נתונים הממוקם על השרת. אם שם הגולש אינו קיים, מנוע ASP רושם את המשתמש החדש במאגר הנתונים, שולח הודעת דואר אלקטרוני לכתובת שהקליד הגולש, ולסיום מכין קובץ HTML שמכיל את ההודעה "Welcome New User" ושולח אותו לגולש.



תרשים 21.4

סיכום

טכנולוגיית ASP מאפשרת בניית יישומים בסביבת Web על ידי מנוע המחולל דפי HTML בזמן ריצה (זמן אמת) מחד, ופקודות גישה למסדי נתונים וביצוע פעולות נוספות מאידך. טכנולוגיה זו תאפשר לנו לייצר דפי HTML עבור כל לקוח באופן המתאים לו ולבקשתו הספציפית, ובכך תקצר באופן משמעותי את זמן כתיבת האתר, מכיון שקטע קוד אחד יכול להיות אחראי למספר רב של דפים.

פרק 22

התקנת מנוע ASP

הפעלה של דפי ASP חייבת להתבצע באמצעות שרת Web, על כך ראה בפרק 21. שרת WEB של Microsoft הוא **IIS** (Internet Information Server). תוכנת שרת זו ניתנת להורדה חינם מתוך אתר הבית של Microsoft כחלק מ-Option Pack4 לבעלי מערכת הפעלה Windows NT או Windows 2000. יש תוכנות המאפשרות הרצת ASP על שרתי UNIX ו-LINUX, כגון chilli!soft, אך לא נדון בהן בספר זה. מכיון שספר זה עוסק בגירסה 3.0 של ASP הנתמכת במלואה רק בשרת האינטרנט של Microsoft בגירסה 5 (IIS5), ישנם אובייקטים ושיטות אשר לא ייתמכו אם תבחרו לעבוד עם שרתים אחרים, כגון Personal Web Server המוסבר בהמשך.

תוכנת **PWS** היא תוכנת שרת שניתן להתקינה על מערכת הפעלה Windows 9x/Me. זו לא תוכנת שרת מלאה כמו **IIS** אבל מספיק טובה כדי ללמוד בעזרתה את העבודה עם ASP.

את תוכנת **PWS** ניתן למצוא בתקליטורים המצורפים לספרים הבאים:

- **HTML 4 למפתחי אתרים באינטרנט**, מהדורה שלישית, זהר עמיהוד, הוצאת **הוד-עמי**.
- **JavaScript למפתחי אתרים באינטרנט**, ירון לייפנברג, הוצאת **הוד-עמי**.
- **JavaScript המדריך השלם**, הוצאת **הוד-עמי**.
- **ASP 3 המדריך השלם**, Stephen Walther , הוצאת **הוד-עמי**.
- **ASP 3 סדנת לימוד**, כולל Visual InterDev, רונן אלמוג, הוצאת **הוד-עמי**.
- **ASP 3 למפתחי אתרים באינטרנט**, ירון ואייל לייפנברג, הוצאת **הוד-עמי**.

Windows 95	התוכנה לא נמצאת בתקליטור ההתקנה. **נדרש לבצע עדכון ל-Winsock 2.0**.
Windows 98	התוכנה נמצאת בתקליטור ההתקנה בתיקיה \add-ons\pws.
Windows Me	התוכנה לא נמצאת בתקליטור ההתקנה.

התקנת Personal Web Server



הערה

הוראות ההתקנה מתייחסות לתקליטור הוד-עמי הנמצא בכל אחד מספרי ההוצאה בנושאי אינטרנט. לספר זה לא מצורף תקליטור. לפרטים ראה בפרק הקדמה.

התקנת Personal Web Server במחשב בו מערכת ההפעלה המותקנת היא Windows 95 דורשת עדכון ל-Winsock 2.0. עליך להתקין את קובץ העדכון W95ws2setup.exe שבתיקיה Software\Upgrade\ שבתקליטור (לספר זה לא מצורף תקליטור. ראה הערה בהקדמה).

לאחר ביצוע העדכון יש לאתחל את המחשב. לאחר שהמחשב "עלה" מחדש:

1. הפעל את הקובץ Setup.exe שבתיקיה Software\PWS\ בתקליטור (ראה הערה בתחילת סעיף זה), או לחילופין במחשב בו פועלת מערכת ההפעלה Windows 98. הפעל את קובץ ההתקנה של Personal Web Server בתיקיה X:\add-ons\pws שבתקליטור ההתקנה המקורי של Windows 98 (החלף את האות X באות המייצגת את כונן התקליטורים שלך).
2. בחלון הפתיחה לחץ על **הבא** (Next).

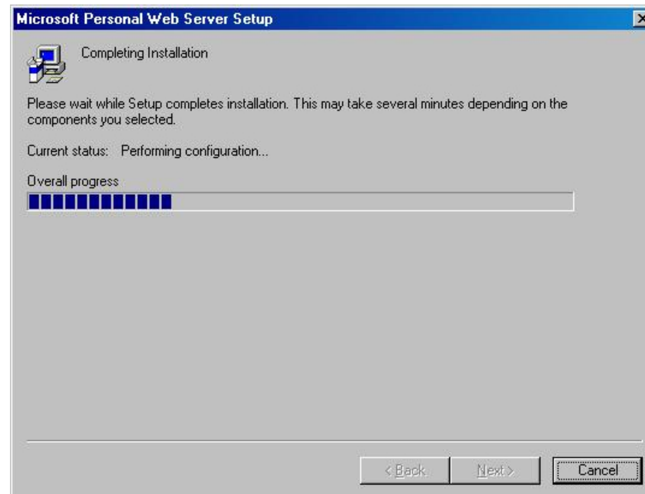


תרשים 22.1

3. בחלון **הסכם השימוש בתוכנה** (EULA) (מופיע בעת התקנה במערכת ההפעלה Windows 95 ולעיתים מופיע ריק) לחץ על **Accept**.

4. בחלון בחירת סוג ההתקנה לחץ על **Typical** (תוכל לקרוא את פירוט אפשרויות ההתקנה בחלון).

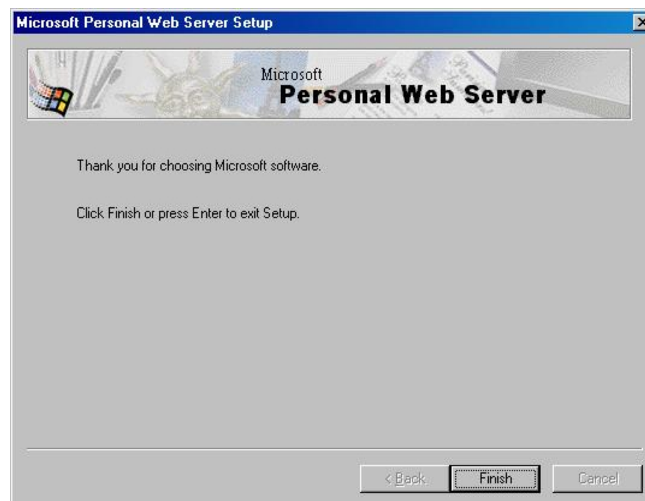
5. בחלון התקנת תיקיית ברירת המחדל לדף הבית לחץ על **הבא** (Next), או שנה את הנתיב לפי רצונך (מומלץ להשאיר את ברירת המחדל כמו שהיא. בהמשך הספר ההתייחסות היא לתיקיית ברירת המחדל, C:\Inetpub\wwwroot).



תרשים 22.2

ההתקנה מתבצעת וקבצים מועתקים למחשב.

6. לסיום ההתקנה יש ללחוץ על **סיום** (Finish) ולאתחל את המחשב.



תרשים 22.3

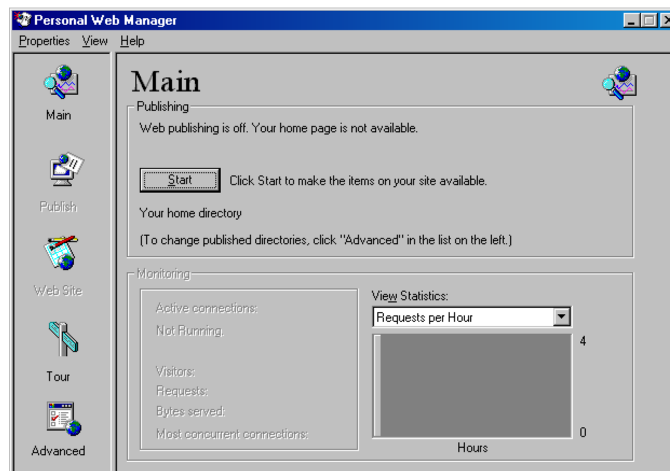
כעת מותקן Personal Web Server במחשב שלך.

הגדרת תצורה של Personal Web Server

בסמוך לשעון בשורת היישומים תוכל להבחין בסמל הבא: . סמל זה מראה ש-Personal Web Server פעיל.

אם סמל זה אינו מופיע, יש להפעיל את Personal Web Server מתוך תפריט: **התחל**>**תוכניות**>**Personal Web Server**>**Personal Web Manager**.

במסך זה יש ללחוץ על לחצן **Start**.



תרשים 22.4

לאחר הלחיצה יופיע המסך.

מעתה יש לאחסן את הקבצים שתיצור בתיקיה **C:\Inetpub\wwwroot** (אלא אם שינית את תיקיית ברירת המחדל לדף הבית, במהלך ההתקנה).

כעת ניגש לבצע בדיקת תקינות.

בסיום ההתקנה יש לאחסן את הקבצים בתיקיה Inetpub\wwwroot. הפנייה אליהם נעשית דרך שורת **כתובת** (address) שם יש לרשום **127.0.0.1/*filename*.asp** ולהקיש **Enter**. <u>אין</u> לפנות לקבצים אלה דרך תפריט **קובץ** (file), **פתיחה** (open) של הדפדפן!!!

בכל פעם שתפנה לקבלת קבצים מהשרת, הם יילקחו מתוך תיקיה זו.

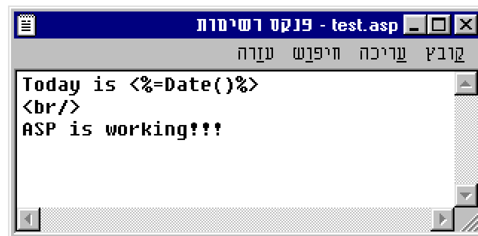
מומלץ לבצע בדיקת תקינות בעזרת תרגיל הבדיקה שבהמשך.

תרגיל בדיקת תקינות

הפעל את פנקס הרשימות והקלד את הקוד הבא:

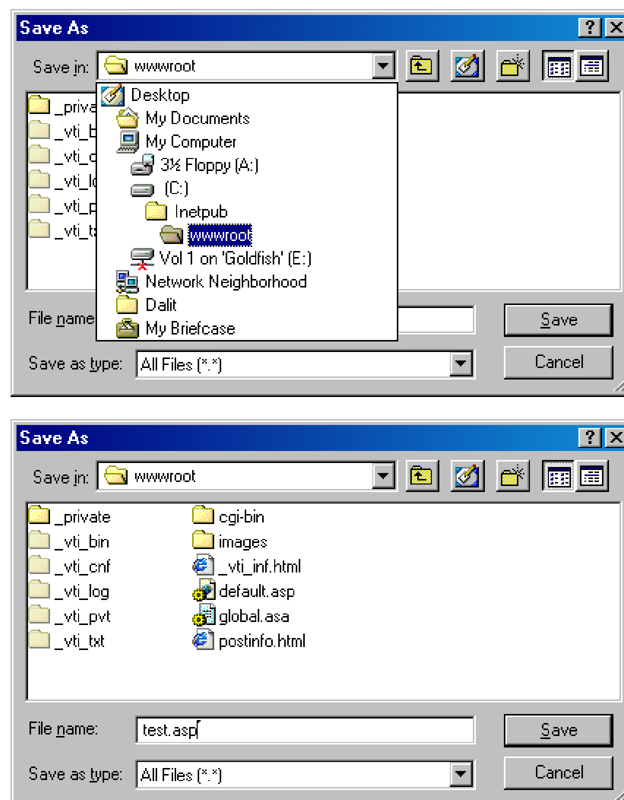
```
Today is <%=Date()%>
<br />
ASP is working!!!
```

הקוד בפנקס הרשימות ייראה כך:



תרשים 22.5

את המסמך יש לשמור כקובץ **test.asp** בתיקיה **C:\Inetpub\wwwroot**.



תרשים 22.6

לאחר שמירת המסמך יש לפתוח דפדפן ולהקליד בשורת הכתובת את הכתובת הבאה, ובסיומה להקיש **Enter**.

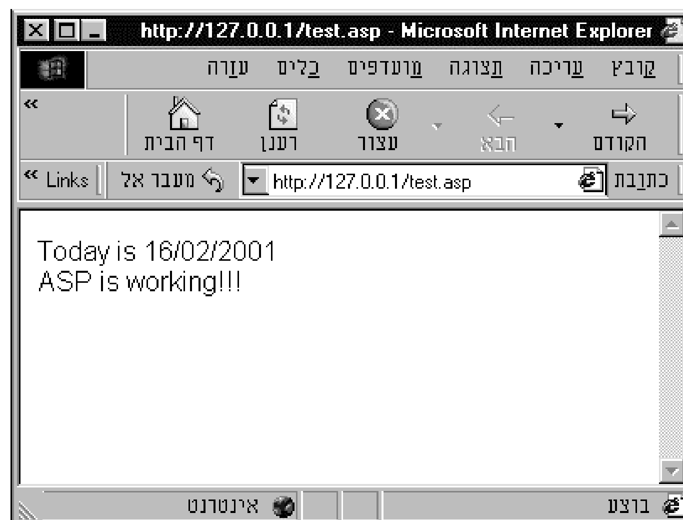
http://127.0.0.1/test.asp

או

http://localhost/test.asp

למעשה, הכתובת 127.0.0.1 שמורה לשימוש גלובלי. כלומר, כל מחשב שיבקש את הכתובת 127.0.0.1 יפנה לעצמו (בתהליך שנקרא **Loop Back**). אם ידועה לך כתובת IP של השרת, מומלץ להשתמש בה במקום 127.0.0.1.

אם כל השלבים הצליחו, התוצאה שתתקבל תיראה כך:



תרשים 22.7

פרק 23

תחביר ושורות קוד ראשונות

בפרק זה נלמד את התחביר הבסיסי אשר יאפשר לנו לבנות את היישומים המתקדמים בהמשך.

התחביר אליו נתייחס בספר זה הוא תחביר VBScript. שפת VBScript מתחלקת לשפת צד **לקוח** (שפה שתעבוד עם הדפדפן במסגרת HTML), ולשפת צד **שרת** אשר תעבוד רק ביישומים אשר מופעלים בשרת. בניגוד לשפת VBScript בצד הלקוח, אשר נתמכת רק בדפדפן Internet Explorer, שפת VBScript בצד השרת מתבצעת ללא קשר לדפדפן. לכן, נוכל לכתוב בשפה זו בלי לדאוג לתאימות מול הדפדפנים השונים.

כדאי לדעת כי ניתן לכתוב קוד בשפות נוספות כגון JavaScript, Perl וכדומה.

ניתן לומר כי ASP היא שפה המתבססת על שפת VBScript, מכיון שיש אובייקטים מיוחדים רק לסביבת ASP. בספר זה לא תתבצע הפרדה זו.

עקרונית, השפה אותה מקבל מנוע ASP כברירת המחדל היא VBScript. אולם, ניתן להכריז על השפה בה משתמשים בקבצי ASP, באופן הבא:

```
<%@LANGUAGE=VBScript%>
<%@LANGUAGE=JavaScript%>
```

שיטה נוספת להכרזת השפה היא:

```
<script language="VBScript" runat="SERVER">
</script>
<script language="JavaScript" runat="SERVER">
</script>
```

מכיון שבספר זה נשתמש רק ב-VBScript לקוד בצד השרת, לא נשתמש בהגדרות המופיעות למעלה.

שפת VBScript אינה Case Sensitive. כלומר, אין משמעות לשימוש באותיות גדולות או קטנות.

ייתכן שבדוגמאות בספר נשתמש באותיות גדולות וקטנות לצרכי הבנה, אך ניתן לכתוב את הפקודות בכל אופן (למעט קטעי קוד הכתובים ב-JavaScript אשר הינה שפה **Case Sensitive**).

כשם שדפדפנים מבצעים קטעי קוד התחומים בין סוגריים זוויתיים (< >), מנוע ASP יודע לבצע קוד התחום בין (<%) ל- (%>). לדוגמה:

```
<%
ASP Code
%>
```

כמו שפות תכנות רבות, גם מנוע ASP מתייחס לקוד על פי שורות ולכן אין לשבור שורות. כלומר, כל שורה מתבצעת בפני עצמה. אם יש צורך לכתוב קוד המתפרס על מספר שורות, יש להשתמש בסימן ה**קו התחתי** (_). לדוגמה:

```
<%
ASP Code
Long ASP Code _
Continues here
%>
```

הצהרת משתנים

משתני VBScript, בדומה למשתני JavaScript, הם משתנים גמישים אשר יכולים לקבל כל סוג ערך מסוג Variant. כלומר, אין צורך להצהיר על סוג המשתנה אלא המשתנה יודע להתאים את עצמו לערך המוזן לתוכו. ליצירת משתנה ניתן להשתמש במילה השמורה DIM. לדוגמה:

```
Dim x
x="hello"
Dim y
y=12
```

יש לציין, כי אין חובה להשתמש במילה DIM. כמו כן, מכיון שאין הגדרה של סוג משתנה (מספר, מחרוזת וכדומה), המשתנה מתאים את עצמו לסוג הערך שהוזן לתוכו. אם מתעורר הצורך להסב סוג משתנה, ניתן לבצע **הסבה** (casting) על ידי שימוש בתחביר:

```
x="41"
y=Cint(x)
```

המשתנה y יקבל את ערכו של x המוסב ממחרוזת למספר. ניתן להסב משתנים לסוגים רבים, כגון:

		טווח
Cint()	החלף לסוג **מספרי** (integer)	32768- עד 32767
Cstr()	החלף לסוג **מחרוזת** (string)	0 עד 2 מיליארד תווים בקירוב
Csng()	החלף לסוג **בודד** (single)	$-3.402823e^{38}$ עד $-1.401298e^{-45}$ / $1.401898e^{-45}$ עד $3.402823e^{38}$
Cdbl()	החלף לסוג **כפול** (double)	$-1.79769313486232e^{308}$ עד $-4.94065645841247e^{-324}$ / $1.79769313486232e^{308}$ עד $4.94065645841247e^{-324}$
Clng()	החלף לסוג **ארוך** (long)	2,147,483,648- עד 2,147,483,647
Cbool()	החלף לסוג **בוליאני** (boolean)	true או false
Cbyte()	החלף לסוג **בית** (byte)	0 עד 255
Ccur()	החלף לסוג **מטבע** (currency)	922,337,203,685,477.5808- עד 922,337,203,685,477.5807
Cdate()	החלף לסוג **תאריך** (date)	1.1.100 עד 31.1.9999 כולל

כמו כן, ניתן להפוך **ייצוג עשרוני** (decimal) לייצוג בבסיס 16 (hexadecimal) על ידי שימוש בתחביר:

```
x=10
y=hex(x)
```

המשתנה y יקבל את הערך A שהוא הייצוג ההקסדצימלי ל- 10.

הסיכוי להשתמש בכל סוגי המשתנים הוא נמוך ביישומי אינטרנט בסיסיים, אך כדאי להכיר את האפשרויות הקיימות.

ASP מחולל דפי HTML. ניתן לחולל קוד דינמי (כגון, משתנים) וכן ניתן להשתמש בקוד סטטי (כגון, שורות טקסט או פקודות HTML). שורות אשר אינן תחומות בין סימני הפתיחה (%>) והסגירה (<%) של ASP, יישלחו אל הלקוח כמו שהן. כלומר, הקובץ:

```
<%
dim x
x = "Mooki"
%>
<font color="red">Welcome</font>
```

יחולל וישלח אל הלקוח את המסמך:

```
<font color="red">Welcome</font>
```

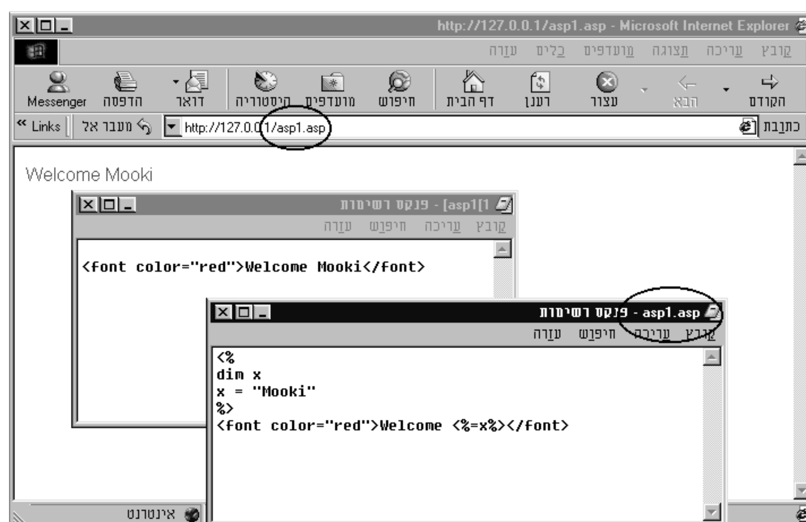
אחת הדרכים לשלב קוד דינמי בשורות הסטטיות היא להשתמש בסימן **שווה** (=).
לדוגמה, אם נרצה לשלול קוד המשלב את שורת HTML הסטטית - <font color="red">Welcome עם ערך המשתנה המוגדר במסגרת קוד ה-ASP, נכתוב את הקוד בצורה הבאה (קובץ **asp1.asp**) :

```
<%
dim x
x = "Mooki"
%>
<font color="red">Welcome <%=x%></font>
```

מסמך ASP זה יחולל וישלח ללקוח את מסמך HTML הבא :

```
<font color="red">Welcome Mooki</font>
```

להלן מסך הדפדפן, הקוד הנראה ב-View Source וקוד ASP :



תרשים 23.1

כלומר, מנוע ASP שלח ללקוח את שורות הקוד הסטטיות (Welcome), ובמקום הקוד הדינמי (<%=x%>) מיקם את ערכו של x. כלומר, Mooki ואחריו את הקוד הסטטי .

עבור הדוגמאות הבאות, כאשר נרצה לשלב ערך משתנה במסמך HTML הנשלח ללקוח, נפתח את הסימן %>, נוסיף את סימן ה**שווה** (=), נרשום את המשתנה ונסגור עם הסימן <%.

הדוגמאות הבאות יכילו את מסך הדפדפן כשמעליו מסמך HTML הנשלח ללקוח (אותו ניתן לראות ב-View Source), ולבסוף את קובץ ASP השמור בשרת. הדגמה זו מראה בצורה ברורה כיצד קוד ASP מתבצע על השרת ומכין מסמך HTML המתבצע על דפדפן הלקוח.

בקרת זרימה ולולאות

כבכל שפת תכנות, גם ב-ASP יש טיפול בבקרת זרימה ובלולאות. מכיון שאנו מניחים שקוראי ספר זה מיומנים בכתיבת JavaScript, אין צורך להסביר את משמעות המושגים, אלא להתמקד בתחביר.

תחביר IF

ב-VBScript ניתן להשתמש בבדיקת **IF** באופן הבא:

```
IF Condition THEN
    ASP Statement
END IF
```

להבדיל מ-JavaScript, התנאי אינו תחום בסוגריים ויש להוסיף את המילה THEN באותה שורה.

כמו כן, הפקודות המתבצעות אם התנאי מתקיים אינן תחומות על ידי סוגריים, אלא מתבצעות עד הפקודה END IF.

ניתן כמובן להשתמש גם במילה **ELSE** באופן הבא:

```
IF x=2 THEN
    Y=Y+1
    Z=Z-8
ELSE
    Y=Y-1
    Z=Z*9
END IF
```

Select Case

בדיקה נוספת (מקבילה לבדיקת Switch ב-JavaScript) נקראת **CASE**, והתחביר לביצוע הבדיקה הוא:

```
SELECT CASE X
CASE 2
    Y=Y+2
CASE 3,4,10
    Y=Y+3
END SELECT
```

המילים SELECT CASE X מגדירות כי הערך הנבחן הוא הערך של המשתנה X. כל שורה הנפתחת במילה CASE מגדירה מקרה בו למשתנה ערך מסוים. לדוגמה, בשורה השנייה מוגדר כי במקרה ולמשתנה X יש ערך 2, יש להוסיף 2 למשתנה Y. בשורה הרביעית מוגדר כי במקרה ולמשתנה X ערך מטווח הערכים המופרדים בפסיקים, יש להוסיף 3 למשתנה Y.

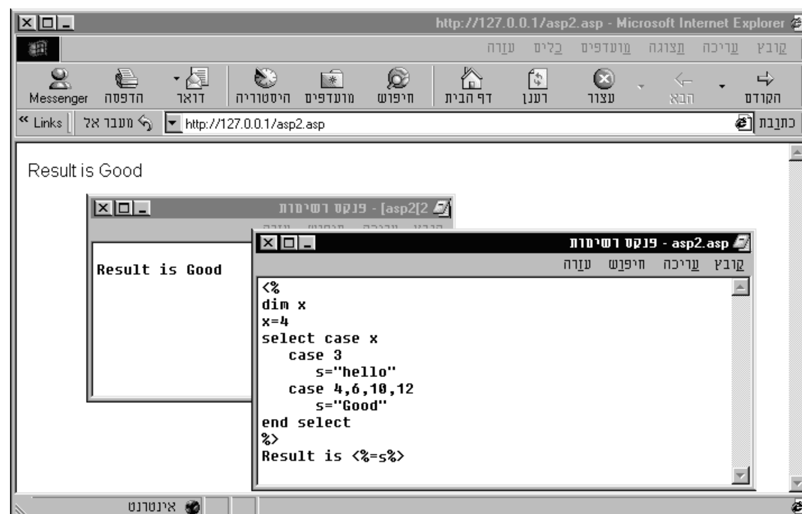
להלן דוגמה (קובץ **asp2.asp**):

```
<%
dim x
x=4
select case x
   case 3
      s="hello"
   case 4,6,10,12
      s="Good"
end select
%>
Result is <%=s%>
```

מסמך HTML שיישלח ללקוח הוא:

```
Result is Good
```

להלן המסך:



תרשים 23.2

לולאת FOR

התחביר לשימוש בלולאת **FOR**, גם הוא שונה מלולאת FOR של JavaScript. בלולאת FOR של VBScript יש להגדיר ערך התחלתי של משתנה, ערך סופי, וניתן להגדיר את השינוי בערך המשתנה באופן הבא:

```
FOR I=1 TO 100 STEP 5
    X=X+1
NEXT
```

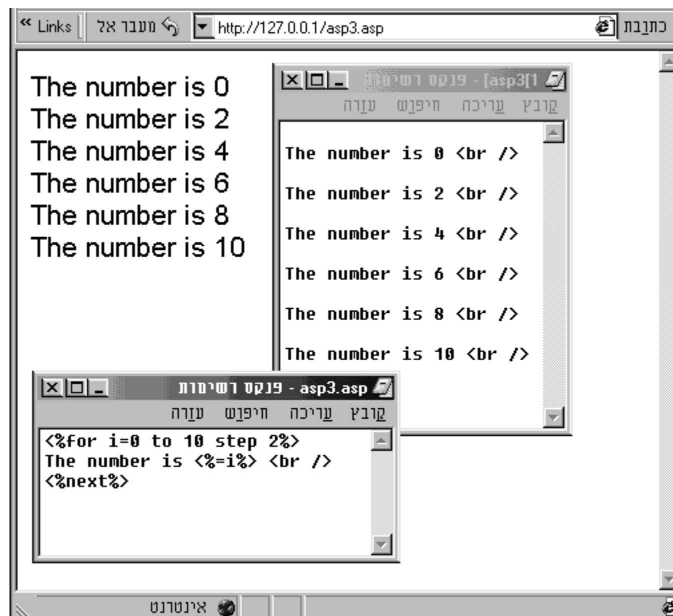
התחביר כולל את המילה FOR, הערך ההתחלתי של המשתנה המונה את ריצות הלולאה, המילה TO והערך הסופי אליו מגיע המשתנה ומפסיק את ריצת הלולאה. המילה STEP מגדירה כי ערכו של המשתנה יעלה ב- 5 בכל ריצה של הלולאה. אם לא נשתמש במילה STEP, ברירת המחדל היא 1. כלומר, ערכו של המשתנה יעלה ב- 1 בכל ריצה של הלולאה. יש לציין כי בספירה שלילית (משתנה המתחיל עם ערך גבוה מערך הסיום) חובה להשתמש במילה STEP להגדרת שינוי הערך.

הגדרת סיום הלולאה היא המילה NEXT.

כדי לראות תוצאה של ריצת לולאה, ניתן ליצור את קובץ ASP הבא (**asp3.asp**) :

```
<%for i=0 to 10 step 2%>
The number is <%=i%> <br />
<%next%>
```

להלן המסך :

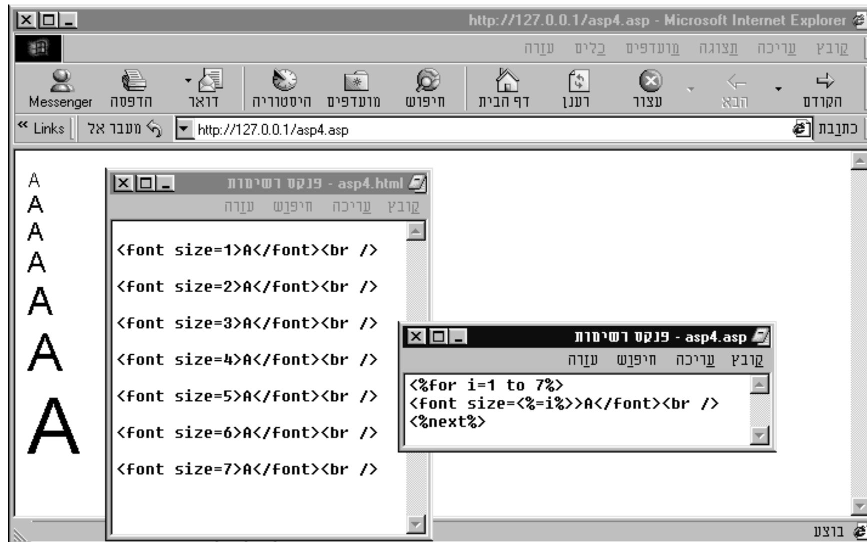


תרשים 23.3

דוגמה נוספת היא (קובץ **asp4.asp**) :

```
<%for i=1 to 7%>
<font size=<%=i%>>A</font><br />
<%next%>
```

להלן המסך:



תרשים 23.4

ניתן לעצור לולאת **FOR** במקרים מסוימים על ידי שימוש בביטוי Exit For. לדוגמה:

```
for I=1 to 10
    if I=7 then
    exit for
    end if
next
```

לולאת DO UNTIL

לולאה זו מאפשרת לבצע ריצות הלולאה עד להתקיימות תנאי מסוים. לדוגמה:

```
DO UNTIL X=2
  X=X-1
LOOP
```

המילה התוחמת את גבולות הלולאה היא LOOP.

לולאת DO WHILE

לולאה זו תתבצע כל עוד מתקיים תנאי מוגדר, כגון:

```
DO WHILE X<2
  X=X+1
LOOP
```

גם בלולאה זו, המילה התוחמת את גבולות הלולאה היא LOOP.

שיטה נוספת לכתיבת לולאה זו:

```
WHILE X<2
   X=X+1
WEND
```

בשיטה זו, המילה התוחמת את תכולת הלולאה היא WEND.

הערה

בכל שפת תכנות ניתן לתעד את הקוד על ידי **הערות** (remarks או comments). כדי לכתוב במסמך שורת הערה שלא תתבצע על ידי מנוע ASP, יש להשתמש בסימן ה**תג** (') באופן הבא:

```
' this is a remark
```

אופרטורים

^	העלאה בחזקה	****	חילוק (התוצאה - מספר שלם)
-	סימן שלילי	**Mod**	שארית (שלמים, Modulo)
*****	כפל	**+**	חיבור
/	חילוק	**-**	חיסור

אופרטור	**דוגמה**	
=	exp1 = exp2	מחזיר true במקרה של שוויון.
<>	exp1 <> exp2	מחזיר true במקרה של אי-שוויון (שונה).
>	exp1 > exp2	מחזיר true כאשר ערך הביטוי משמאל גדול מערך הביטוי מימין.
>=	exp1 >= exp2	מחזיר true כאשר ערך הביטוי משמאל גדול מערכו או שווה לערך הביטוי מימין.
<	exp1 < exp2	מחזיר true כאשר ערך הביטוי משמאל קטן מערך הביטוי מימין.
<=	exp1 <= exp2	מחזיר true כאשר ערך הביטוי משמאל קטן מערכו או שווה לערך הביטוי מימין.

אופרטור	משמעות	דוגמה
NOT	שלילת הביטוי.	NOT a
AND	בודק האם שני הביטויים הם True.	a AND b
OR	בודק האם לפחות אחד הביטויים הוא True.	a OR b
XOR	בודק האם אחד ורק אחד מהביטויים הוא True.	a XOR b
EQV	בודק אם לשני הביטויים יש אותו ערך.	a EQV b
IMP	בודק אם יש לביטוי משמעות לוגית.	a IMP b

מערכים

הכרזה על מערך חד-מימדי נעשית על ידי שימוש בפונקציה **Array()**:

```
MyArray = Array ("yossi", "Benny", "meshulam", "Dalit")
```

פנייה לאיבר במערך נעשית על ידי ציון מיקומו בתוך סוגריים מעוגלים (האיבר הראשון נמצא במיקום 0):

```
MyArray(3) = "Mendelbaoum"
```

ניתן להגדיר את גודל המערך ללא הזנה ראשונית של ערכים, על ידי שימוש בהכרזת Dim וציון מספר האיברים הייעודי:

```
Dim MyArray(6)
```

כדי ליצור מערכים רב-מימדיים יש להשתמש בהכרזת Dim, ציון מספר המימדים וגודלם. כך לדוגמה, הכרזה על מערך דו-מימדי בעל 3 איברים במימד הראשון ו- 2 איברים בשני, תיראה כך:

```
Dim MyArray(3,2)
```

בשימוש במערכים יש חובה לציין את גודלם, אולם לעיתים גודלו של המערך יכול להשתנות.

במקרים בהם נרצה להגדיל או להקטין את המערך יש להשתמש בהכרזת ReDim, המאפשרת לנו להקצות בצורה דינמית גודל חדש למערך. כך לדוגמה, מערך שגודלו הראשוני היה של 5 איברים יגדל בעזרת ReDim ל- 10 איברים:

```
Dim MyArray(5)
ReDim MyArray(10)
```

חשוב לציין, כי שינוי גודל המערך ימחק את תכולתו הקודמת, למעט אם נשתמש בהכרזת Preserve המורה לשמור את תכולת המערך:

```
Dim MyArray(10)
ReDim Preserve MyArray(20)
```

פרק 24

אובייקט Request – התחלת הקשר בין השרת ללקוח

בסוף פרק זה נדע לקבל מידע מהלקוח (דרך טופס HTML), ולהחזיר תשובה דינמית.

ל-ASP מספר אובייקטים המאפשרים לבנות יישומים אינטראקטיביים בין השרת ללקוח. בהמשך נבין את משמעות האובייקטים השונים ואף נשתמש בהם. האובייקט הראשון, ואחד המשמעותיים ביותר, הוא **אובייקט Request**. אובייקט זה מאפשר לקבל מידע מדפדפן הלקוח החל בנתונים המוזנים לתוך שדות בטופס וכלה ב-cookies ונתונים נוספים. כדי להשתמש באובייקט זה, יש להבין את פעולת טפסי HTML.

הבנת טפסי HTML

טופס HTML מורכב ממספר אובייקטים. אובייקט מסוג אחד הוא שדה הקלט (text, button, checkbox, radio...) המאפשר קליטת נתונים מהמשתמש. האובייקט החשוב הבא הוא אובייקט ה**הגשה** (submit) המיוצג כלחצן, אשר עם לחיצתו נשלחים הנתונים שנאספו בטופס.

מכיון שלחיצה על לחצן **submit** שולחת את הנתונים, יש להגדיר בצורה מדויקת להיכן נשלחים הנתונים. הגדרה זו מתבצעת באובייקט פתיחת הטופס (<form>) על ידי התכונה **action**. תכונה זו מגדירה לאיזה יישום ולאיזה שרת לשלוח את נתוני הטופס, והגדרתה מתבצעת באופן הבא:

```
<form action="http://www.yogli.com/search_application">
```

נתוני **טופס** (FORM) נשלחים אל השרת באחת משתי דרכים, post או get.

Post

שיטה זו היא השיטה הטבעית לשליחת נתונים על גבי פרוטוקול HTTP. כדי לשלוח נתוני טופס ב-**Post**, יש להוסיף בשורת הגדרת הטופס את המילים "method="post באופן הבא:

```
<html>
<head>
</head>
<body>
<form action="http://www.mysite.com" method="post">
   Type in your name:
   <input type="text" name="x" />
   <input type="submit" />
</form>
</body>
</html>
```

טופס זה ייראה כך בדפדפן:

תרשים 24.1

בעת לחיצה על לחצן **Submit Query** (או לחצן **שליחת שאילתה**), המשתנה x יישלח לשרת המוגדר ב-action כשערכו יהיה המלל שהוזן לתוך שדה הטקסט. כלומר, אם המשתמש הזין לשדה את המלל 'Hello' ולחץ על לחצן **Submit Query**, המשתנה x וערכו יישלחו לשרת www.mysite.com בצורה הבאה: x=Hello.

Get

שיטה זו הינה שיטה חילופית לשליחת מידע לשרת, והיא נבדלת משיטת Post בכך שכמות המידע שניתן להעביר בה היא מוגבלת, וכן ניתן לראות את הנתונים בשורת הכתובת. כלומר, הנתונים יוצגו בהמשך לכתובת השרת. לדוגמה:

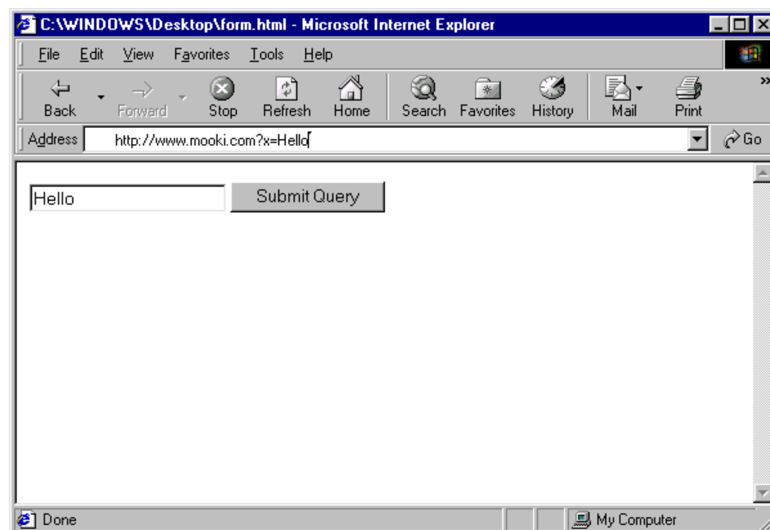
http://www.mooki.com?x=Hello

המשתנים וערכיהם מופרדים מכתובת השרת על ידי **סימן שאלה** (?). אם יש יותר ממשתנה אחד, המשתנים יופרדו על ידי סימן **אמפרסנד** (&), באופן הבא:

http://www.mooki.com?x=Hello&y=Kriger

כדי לשלוח את נתוני הטופס בשיטת **Get**, יש להוסיף לפקודת Form את המילים method="get", אם כי ברירת המחדל של הדפדפן היא לשלוח את הנתונים ב-Get אם לא נקבע אחרת.

כך תיראה שליחת נתונים ב-Get. שים לב לשורת הכתובת:

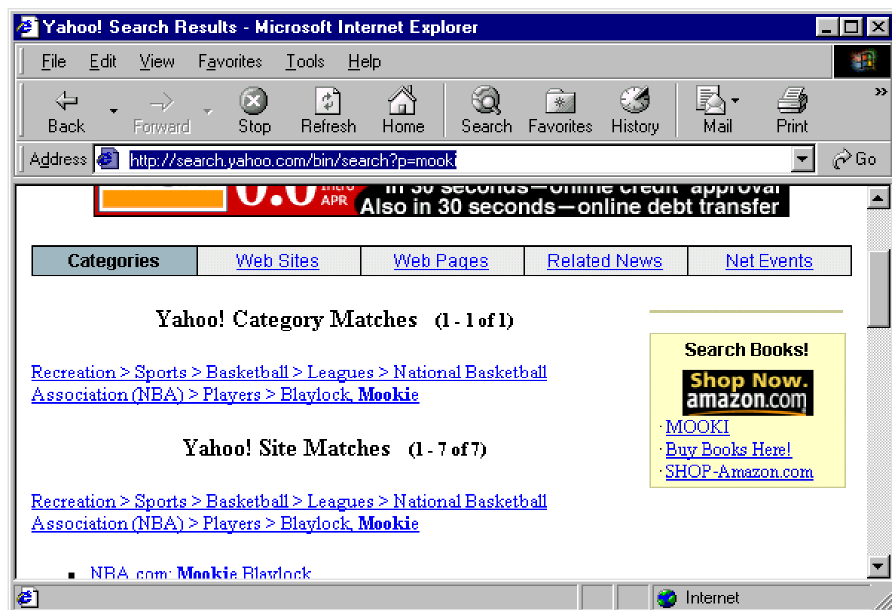


תרשים 24.2

אם כן, קיימות שתי דרכים לשליחת הנתונים לשרת. לכל דרך יתרונות וחסרונות ויש להחליט לגבי כל יישום באיזו שיטה להשתמש. מנועי חיפוש מעדיפים להשתמש בשיטת Get, מכיון שעל שורת הכתובת המכילה את המשתנים המייצגים את החיפוש ניתן להגדיר **מועדפים** (Favorites), ובכך לשמור חיפושים מוצלחים.

לדוגמה, משתמש ירצה לשמור את הגדרת החיפוש הבאה:





תרשים 24.3

בניית מסמך HTML המאפשר חיפוש באתר של YAHOO

למעשה, ניתן לכתוב מסמך HTML המאפשר לשלוח שאילתות לאתר הבית של YAHOO.

היישום המבצע את החיפוש באתר הבית של YAHOO נקרא **search** והוא נמצא בכתובת:

http://search.yahoo.com/bin/search

יש להכניס כתובת זו לתכונת action של הטופס, כך שמילות החיפוש יישלחו אליה באופן הבא:

```
<form action="http://search.yahoo.com/bin/search">
```

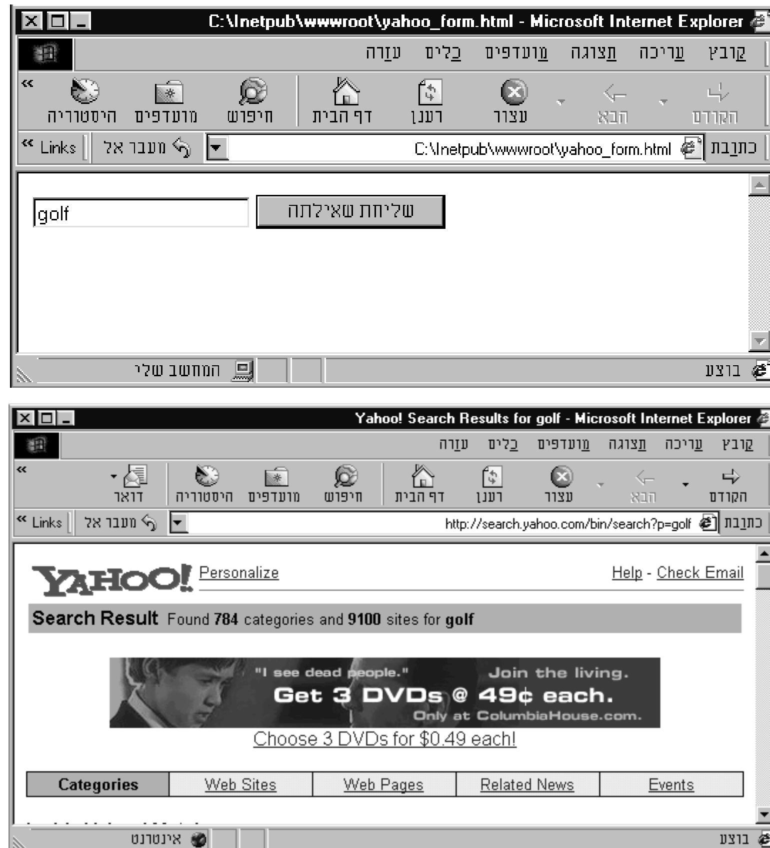
לאחר מכן יש לבנות שדה טקסט בעל השם p, מכיון שמנוע החיפוש של YAHOO ממתין למשתנה בשם זה:

```
<input type="text" name="p" />
```

לפניך קוד HTML המלא השולח מילת חיפוש למנוע החיפוש של YAHOO (**yahoo_form.html**):

```
<html>
    <head>
        <title>
        </title>
    </head>
    <body>
        <form action="http://search.yahoo.com/bin/search" method="get">
            <input type="text" name="p" />
            <input type="submit" />
        </form>
    </body>
</html>
```

להלן המסכים :



תרשים 24.4

כמובן, ניתן לייצר את שורת הכתובת באופן ידני כשמדובר בשיטה Get. לעומתה, מהווה השיטה Post אמצעי מאובטח יותר להעברת מידע, מכיון שלא ניתן לראות את המשתנים בשורת הכתובת.

תרגיל ראשון בתקשורת בין שרת ללקוח

תחילה נבנה טופס HTML רגיל. בפקודת form נכניס ל-action את כתובת השרת עליו אנו עובדים, כלומר **127.0.0.1** (ראה תרגיל בדיקת תקינות בפרק 22). אנו מעוניינים כי הנתונים שיוזנו לטופס יישלחו לקובץ ASP אותו נבנה בהמשך, אשר ייקרא **get_form.asp**. אם כך, שורת הקוד הראשונה במסמך HTML תהיה :

```
<form action="http://127.0.0.1/get_form.asp" method="get">
```

שים לב כי נשתמש בשיטה **Get**.

נוסיף את המלל:

```
Type in your name:
```

לאחר מכן נוסיף שדה טקסט בשם x.

```
<input type="text" name="x" />
```

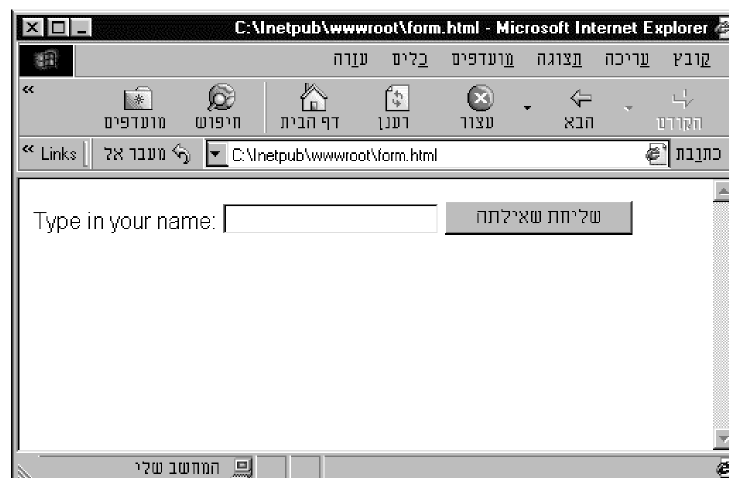
וכן נוסיף לחצן **Submit** ונסגור את הטופס באופן הבא:

```
<input type="submit" />
</form>
```

הקובץ **form.html** ייראה כך:

```
<html>
    <head>
        <title>
        </title>
    </head>
    <body>
        <form action="http://127.0.0.1/get_form.asp" method="get">
            Type in your name:
            <input type="text" name="x" />
            <input type="submit" />
        </form>
    </body>
</html>
```

להלן המסך של **form.html**:



תרשים 24.5

כעת נבנה מסמך ASP בשם **get_form.asp**.

לאובייקט **Request** מספר שיטות אשר יפורטו בהמשך, אך כרגע נתייחס לשיטה הראשונה: **QueryString()**. למעשה, QueryString מוגדרת כתכונה, אך בשל העובדה שיש להעביר אליה ערכים, נתייחס אליה בספר זה כאל שיטה.

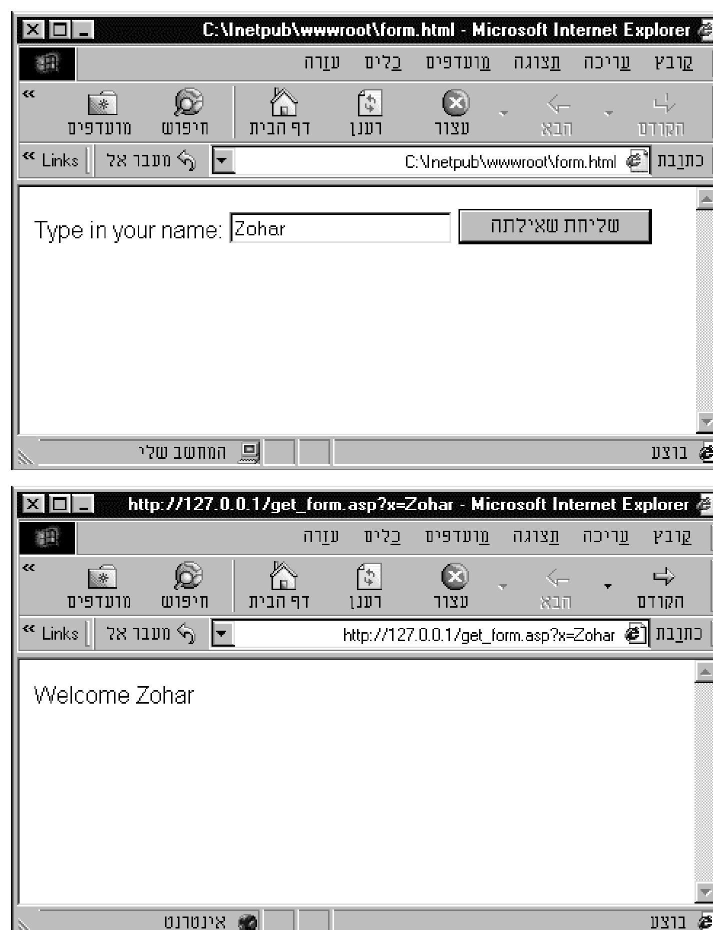
השיטה QueryString() מקבלת את מחרוזת המשתנים הנשלחת בשיטה Get. יש לציין איזה משתנה אנו מעוניינים לקבל, ומכיון שהמשתנה הנשלח מטופס HTML שבנינו נקרא x, אם נרצה לקבל את ערכו בטופס HTML נצטרך לכתוב:

```
request.querystring("x")
```

בהדגמה פשוטה זו נכתוב קובץ ASP בשם **get_form.asp**, ובו נברך Welcome אליה נצרף את ערך המשתנה x אשר הגיע מהטופס, באופן הבא:

```
Welcome <%=Request.QueryString("x")%>
```

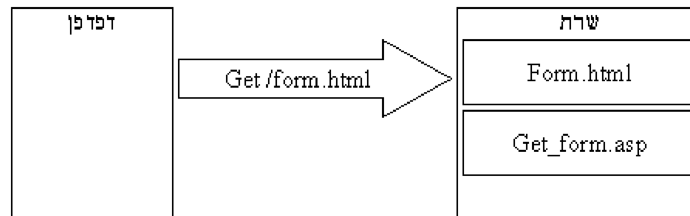
להלן המסכים של התרגיל:



תרשים 24.6

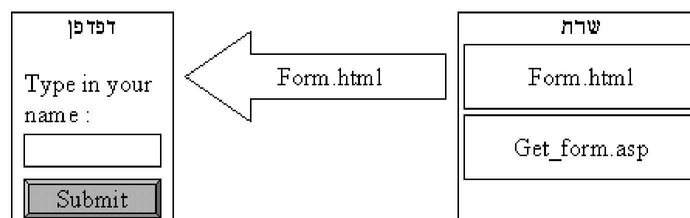
הבה נבין את התהליכים המתרחשים בתרגיל זה :

שלב א



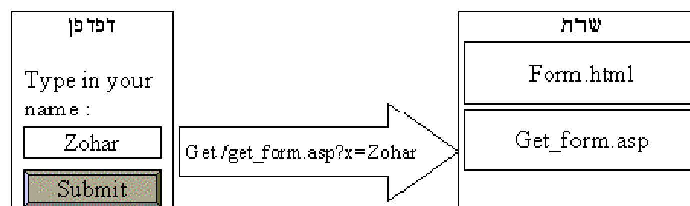
הדפדפן מבקש את מסמך HTML (**form.html**).

שלב ב



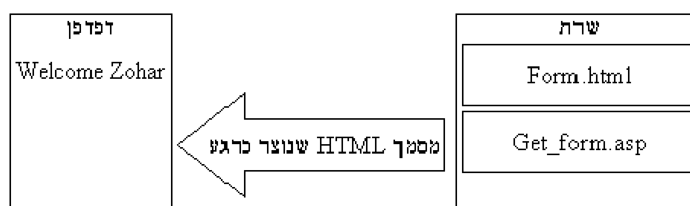
השרת שולח את מסמך HTML ללקוח. הדפדפן מבצע את קוד HTML שקיבל ומציג מלל, שדה טקסט ולחצן **Submit**.

שלב ג



המשתמש מקליד את שמו בשדה הטקסט ולוחץ על לחצן **Submit**. ברגע הלחיצה, אוסף הטופס את ערך השדה, משרשר אותו לכתובת המופיעה בהגדרת Action ושולח.

שלב ד



קובץ ASP מקבל את הפנייה ומכין מסמך HTML חדש, המורכב מהמלל Welcome ומהערך של משתנה x. המסמך החדש נשלח שוב ללקוח.

כרטיס ברכה אישי ב-ASP

קטעי הקוד הבאים יאפשרו לגולש להגדיר לעצמו כרטיס ברכה אשר יכלול צבעים שונים, גודל וצבע גופנים ואף תוכן מילולי דינמי.

הצעד הראשון יהיה לבנות מסמך HTML בשם **personal_card.html**:

```
<html>
    <head>
        <title>
        </title>
    </head>
    <body bgcolor="teal" text="white">
        <p align="center">
            <font size="7">
            Personal Greeting Card Generator
            </font>
        </p>
        <hr />
        Welcome to the personal greeting card generator. In this application you can
        create <br />
        Your own greeting card by determaining the background color, font color &amp;
        size and <br />
        The greeting's message.
        <hr />
```

לאחר מכן, נבנה טופס אשר ישלח את הנתונים לקובץ ASP שייבנה בשלב הבא, וייקרא **create_card.asp** באופן הבא:

```
<form action="http://127.0.0.1/create_card.asp" method="get">
```

כעת נוסיף שדות אשר יאפשרו לגולש להגדיר את צבע הרקע של כרטיס הברכה, צבע וגודל המלל ותוכן הברכה.

```
Choose a background color :
    <select name="bgcolor">
        <option value="red">red</option>
        <option value="yellow" selected>yellow</option>
        <option value="blue">blue</option>
        <option value="olive">olive</option>
        <option value="green">green</option>
    </select>
    <br />
```

```
Write a font color :
<input type="text" name="fontcolor" value="black" />
<br />
Choose a font size :
<select name="fontsize">
    <option value="1">1</option>
    <option value="2">2</option>
    <option value="3">3</option>
    <option value="4">4</option>
    <option value="5">5</option>
    <option value="6">6</option>
    <option value="7" selected>7</option>
</select>
<br />
Type in your message:
<textarea name="cardmessage">Type in your text
</textarea>
<br />
```

לאחר מכן, נוסיף לחצן **Submit** ונסגור את הטופס.

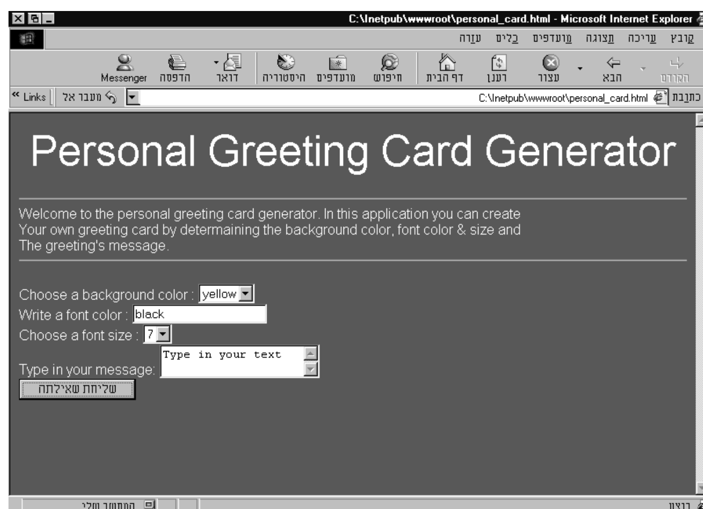
```
<input type="submit" />
</form>
```

כך ייראה הקוד המלא של **personal_card.html**:

```
<html>
    <head>
        <title>
        </title>
    </head>
    <body bgcolor="teal" text="white">
        <p align="center">
            <font size="7">
            Personal Greeting Card Generator
            </font>
        </p>
        <hr />
        Welcome to the personal greeting card generator. In this application you can
        create <br />
        Your own greeting card by determaining the background color, font color &amp;
        size and <br />
        The greeting's message.
        <hr />
        <form action="http://127.0.0.1/create_card.asp" method="get">
            Choose a background color :
            <select name="bgcolor">
                <option value="red">red</option>
                <option value="yellow" selected>yellow</option>
```

```
            <option value="blue">blue</option>
            <option value="olive">olive</option>
            <option value="green">green</option>
        </select>
        <br />
        Write a font color :
        <input type="text" name="fontcolor" value="black" />
        <br />
        Choose a font size :
        <select name="fontsize">
            <option value="1">1</option>
            <option value="2">2</option>
            <option value="3">3</option>
            <option value="4">4</option>
            <option value="5">5</option>
            <option value="6">6</option>
            <option value="7" selected>7</option>
        </select>
        <br />
        Type in your message:
        <textarea name="cardmessage">Type in your text
        </textarea>
        <br />
        <input type="submit" />
    </form>
  </body>
</html>
```

וכך נראה המסך:



תרשים 24.7

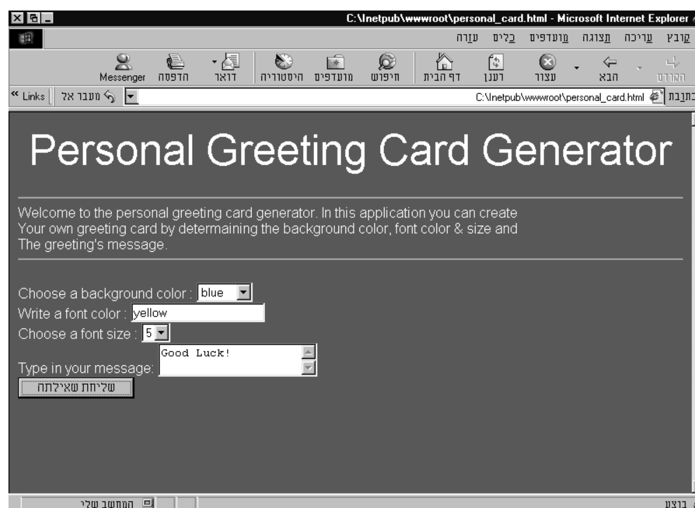
כעת, נכתוב את קובץ ASP בשם **create_card.asp** אשר יקבל את הנתונים מהטופס ויבנה באופן דינמי את כרטיס הברכה עבור המשתמש. בקובץ זה נשתמש ב-request.querystring עבור כל אחד מהפרמטרים שיישלחו מהטופס. להבהרה, אם בטופס נבחר **blue** עבור צבע הרקע, נקליד **yellow** עבור צבע הטקסט, נבחר במספר **5** עבור גודל הגופן ובתיבת בטקסט נקליד **Good Luck!** ונלחץ על לחצן **שליחת שאילתה**. ההודעה הבאה תישלח אל השרת:

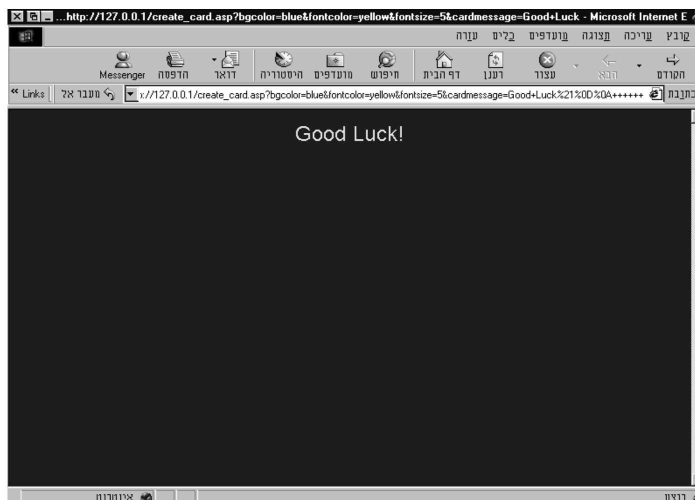
http://127.0.0.1/create_card.asp?bgcolor=blue&fontcolor=yellow&fontsize=
5&cardmessage=Good+Luck%21%0D%0A++++++

כעת, כל שנותר הוא לשלב את פקודות HTML עם ערכי הטופס באופן הבא. להלן התוכן של הקובץ **create_card.asp**:

```
<html>
<head>
</head>
<body bgcolor="<%=request.querystring("bgcolor")%>"
text="<%=request.querystring("fontcolor")%>">
<p align="center">
<font size="<%=request.querystring("fontsize")%>">
<%=request.querystring("cardmessage")%>
</p>
</font>
</body>
</html>
```

להלן המסכים של יישום זה בפעולה:





תרשים 24.8

עוד על אובייקט Request

כעת אנו יודעים לשלוח נתונים לקובץ ASP ולהשתמש בהם. כעת נכיר את אובייקט Request לעומק. לאובייקט זה, כמו לאובייקטים אחרים, יש מספר שיטות ומאפיינים בהם ניתן להשתמש.

השיטה הראשונה בה השתמשנו היא ()querystring. שיטה זו משמשת לקבלת ערכי משתנים המגיעים מטופס בשיטת Get. אם הנתונים נשלחים בשיטת Post ניאלץ להשתמש בשיטה המקבילה ()Form. אין הבדל בין הפונקציונליות של שתי השיטות. לקבלת משתנה x שנשלח ב-Get נכתוב:

```
request.querystring("x")
```

ולקבלת אותו משתנה שנשלח ב-Post נכתוב:

```
request.form("x")
```

cookies

תכונה זו של אובייקט Request מאפשרת שליפה מקבצי **Cookies** של הלקוח, ועל כך נדון בפרק 28.

ServerVariables()

שיטה נוספת של אובייקט request, היא **ServerVariables()** המאפשרת להציג 43 פריטי מידע על הלקוח ועל השרת.

לדוגמה, ניתן לקבל את כתובת IP של הגולש על ידי כתיבת הקוד הבא:

```
<%
Your IP is <%=request.servervariables("remote_addr")
%>
```

להלן חלק מהנתונים שניתן לקבל בעזרת ()request.servervariables:

- request.servervariables("all_http")

תכונה זו מאפשרת גישה ל-HTTP Header הנשלח מהלקוח. HTTP Header הוא הקדמה למסמך HTML הנשלח ללקוח וכן למידע המגיע ממנו, ובו מוגדרים נתונים כגון השפה בה משתמש הלקוח, הדפדפן הספציפי ועוד. ב-HTTP Header ניתן למצוא מידע חשוב על הלקוח הגולש. להלן דוגמה ל-HTTP Header:

```
HTTP_ACCEPT:*/*
HTTP_ACCEPT_LANGUAGE:he
HTTP_CONNECTION:Keep-Alive
HTTP_HOST:127.0.0.1
HTTP_USER_AGENT:Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)
HTTP_COOKIE:ASPSESSIONIDFFEKPMTT=BPHPPECCPDMAHCEHIJCHIDEB
HTTP_ACCEPT_ENCODING:gzip, deflate
```

- Request.servervariables("appl_physical_path")

תכונה זו מאפשרת לקבל את התיקיה האמיתית בה נמצאים הקבצים.

- Request.servervariables ("content_length")

תכונה זו תחזיר את אורך המסמך שהתקבל מהלקוח.

- Request.servervariables ("request_method")

תכונה זו קובעת אם הגולש שולח את הנתונים ב-Post או ב-Get.

לולאת for each

לולאה מיוחדת בה ניתן להשתמש היא לולאת **for each**. לולאה זו בנויה לרוץ על רשימת עצמים (אובייקטים) מוגדרת מראש הנקראת **אוסף** (collection), ולהתייחס לכל פריט בנפרד.

לדוגמה, אם נפנה אל מסמך ASP ונשלח ב-GET את הפרמטרים הבאים: x=1&y=2&z=3, נוכל לקבוע לולאת for each אשר תחלץ את הערך מכל משתנה.

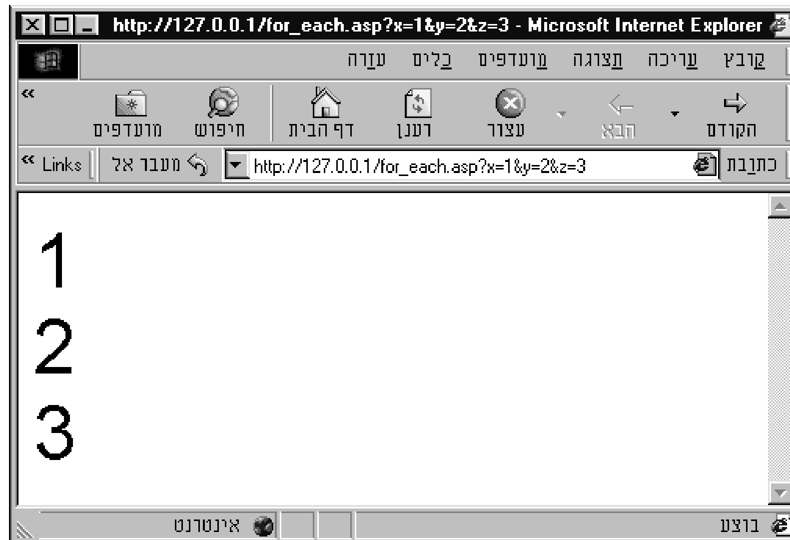
לדוגמה, התחביר של קובץ **start_for_each.html**:

```
<html>
    <head>
        <title>
        </title>
    </head>
    <body>
        <form action="http://127.0.0.1/for_each.asp" method="get">
            <input type="text" name="x" value="1" /><br />
            <input type="text" name="y" value="2" /><br />
            <input type="text" name="z" value="3" /><br />
            <input type="submit" />
        </form>
    </body>
</html>
```

להלן התחביר של קובץ asp בשם **for_each.asp**, הקובע כי עבור כל **פריט** (item) **באוסף** (querystring) יוצג ערכו:

```
<html>
    <head>
        <title>
        </title>
    </head>
    <body>
      <font size="7">
        <%for each item in request.querystring()%>
            <%=request.querystring(item)%>
            <br />
        <%next%>
      </font>
    </body>
</html>
```

אם נקרא לקובץ זה עם הפרמטרים בשורת הכתובת (שים לב לשורת הכתובת), נקבל את התוצאה הבאה :



תרשים 24.9

דוגמה נוספת ללולאת for each נמצאת בפרק 26.

פרק 25

אובייקט Response

מהו אובייקט ASP?

ל-**ASP 3.0** יש 7 אובייקטים. אובייקטים אלה מכילים קטעי קוד מוכנים (שיטות, methods) המאפשרים למפתח לבצע פעולות מורכבות בשורות קוד בודדות. כפי שנלמד בפרק הקודם, כדי לקבל משתנה הנשלח מטופס יש לכתוב ("request.querystring("x בלבד. כך ניתן להשתמש בכל אחד מהאובייקטים לביצוע מטלות שונות. האובייקטים המוגדרים ב-ASP הם:

request
response
application
session
server
object context
asp error

בפרק זה נכיר את אובייקט **Response**.

Response

כשם שאובייקט Request מאפשר קבלת נתונים, אובייקט **Response** משמש לשליחת נתונים אל הלקוח. בפרק זה נסקור חלק מה**שיטות** (methods) וה**תכונות** (properties) של אובייקט זה.

Write()

השיטה הראשונה אותה נסקור היא השיטה **Write()**, המאפשרת כתיבה לקובץ HTML הנשלח אל הלקוח. כלומר, במקום לרשום:

```
<%
dim x
x = "hello"
%>
Welcome <%=x%>
```

ניתן לרשום:

```
<%
dim x
x="hello"
response.write "Welcome " & x
%>
```

שרשור

שים לב, ששרשור המחרוזות מתבצע בעזרת סימן **אמפרסנד** (&). כשם שב-JavaScript אנו משרשרים מחרוזות טקסט בעזרת סימן החיבור (+), נשרשר ב-ASP בעזרת &. ניתן לשרשר ב-ASP גם בעזרת סימן החיבור (+), אך בשל מצבים אשר מכילים מספרים ובהם עלול סימן החיבור לבצע חיבור מתמטי, נשתמש ב**אמפרסנד** (&).

להבנה מלאה של Response.write(), מומלץ לנסות לבצע את תרגיל כרטיס הברכה מהפרק הקודם בשיטה זו, ולשנות אותו לפי הרשום בהמשך.

קובץ **personal_card3.html** הוא העתק מדוייק של קובץ **personal_card.html**, למעט המשפט הבא:

```
<form action="http://127.0.0.1/create_card3.asp" method="get">
```

להלן קובץ **create_card3.asp** המחליף את הקובץ **create_card.asp** בו השתמשנו בפרק הקודם:

```
<html>
<head>
</head>
<%
response.write "<body bgcolor='" & request.querystring("bgColor") &_
"' text='" & request.querystring("fontcolor") & "'>" &_
"<p align='center'>" &_
"<font size='" & request.querystring("fontsize") & "'>" &_
request.querystring("cardmessage")
%>
</p>
</font>
</body>
</html>
```

התוצאה זהה לתוצאה בפרק הקודם.

שים לב למספר נקודות:

שורות קוד ב-ASP יש לכתוב עד להקשת Enter. במידה ורוצים "לפצל" את השורה הארוכה למספר שורות, יש לעשות זאת בעזרת קו תחתי. זהו עניין עיצובי בלבד ואין לזה קשר לביצוע.

ערכי המאפיינים בתגית HTML צריכים להיות בין גרשיים או בין גרש בודד. שרשור המחרוזת, שבדוגמה שלעיל, עושה שימוש בגרשיים כך שניאלץ להשתמש בגרש בודד. את הגרש הבודד הדגשנו למענך. ולידיעה, גרש אחד קטן שלא יהיה במקומו והדפדפן כבר יודיע לך...

Redirect

שיטה זו מאפשרת להפנות לקוח לאתר או דף אחר. בדוגמה הבאה, הלקוח מזדהה דרך טופס טקסט. במידה והסיסמה היא "eggnog", הלקוח מופנה לדף **users.html**. בכל מקרה אחר, יקבל הלקוח את עמוד הפתיחה.

תחילה נבנה טופס HTML בשם **authentication.html**:

```
<html>
    <head>
        <title>
        </title>
    </head>
    <body bgcolor="teal" text="white">
        Please type in your password name:
        <form action="http://127.0.0.1/check_authentication.asp" method="post">
            <input type="password" name="p" />
            <input type="submit" />
        </form>
    </body>
</html>
```

כעת, נבנה קובץ ASP המקבל את הערך של p ובודק אם הוא שווה ל-eggnog. במידה וכן, תתבצע הפניה לדף **users.html**. במידה והסיסמה לא נכונה, יוצג דף על רקע צהוב. שם הקובץ **check_authentication.asp**:

```
<%
if Request.Form("p")="eggnog" then
    Response.Redirect "users.html"
  else
    Response.Write "<html><body bgcolor='yellow'><font size='4'>" &_
    "Welcome to the guest area</font></body></html>"
end if
%>
```

נוסיף גם את המסמך **users.html** כדי לסיים את ההדגמה:

```
<html>
  <head>
    <title>
    </title>
  </head>
  <body bgcolor="pink">
      <font size="5">Welcome true user!</font>
  </body>
</html>
```

חשוב לציין כי פעולת **Redirect** יכולה להתבצע רק לפני שנשלח משהו ללקוח. כלומר, לא ניתן לבצע הפניית Redirect, אם התבצע ()response.write כלשהו או נכתב קוד HTML (למעט מקרה של שימוש בתכונה Buffer אשר נידונה בהמשך פרק זה).

Cookies

תכונה זו מאפשרת שליחת cookies ונידונה בהרחבה בפרק 28.

End

ברגע שמנוע ASP מזהה את המשפט response.end הוא מפסיק את שליחת הנתונים ללקוח. כלומר, בדוגמה הבאה לא תתבצע השורה השלישית.

```
<%
response.write "Will appear"
response.end
response.write "Will not appear"
%>
```

Page Reentry

זו נקודה טובה להציג את אחת התפיסות המקובלות ב-Microsoft לכתיבת קוד. בכל הטפסים שכתבנו עד כה הגדרנו את יעד שליחת הנתונים במאפיין action. אם נגדיר את יעד שליחת הנתונים לאותו קובץ (**Reentry1.asp**), יישלחו הנתונים לאותו הקובץ. כעת, ניצור קובץ ASP אשר מציג טופס HTML וגם יודע לקבל את פרטיו. בשלב ראשון יוצג הטופס בכל פעם שעולה הדף (קובץ **Reentry1.asp**) :

```
<html>
    <head><title></title></head>
    <body>
        <form action="http://127.0.0.1/Reentry1.asp" method="get">
            <input type="text" name="x" />
            <input type="submit" />
        </form>
        <hr />
        The last word submitted was <%=Request.QueryString("x")%>
    </body>
</html>
```

מסמך זה יעלה בפעם הראשונה כך :

תרשים 25.1

שים לב, שמכיון שאין עדיין ערך לשדה x, אין מילה כלשהי שמופיעה לאחר המשפט :

The last word submitted was

כעת נמלא את השדה ונלחץ על לחצן **Submit**. מכיון שאין הגדרת יעד לשליחת הטופס, הנתונים יישלחו שוב ל-**Reentry1.asp** והתוצאה תיראה כך :

תרשים 25.2

השימוש המעשי של תפיסה זו הוא לרכז את הטופס ואת התגובה לטופס באותו מסמך.

בקובץ הבא נגדיר טופס HTML :

```
<form>
        Type in your name: <input type="text" name="x" />
        <input type="submit" />
</form>
```

מכיון שאין אנו מעוניינים להציג את הטופס שוב לאחר שליחת הנתונים, נבדוק אם הנתונים הוקלדו באמצעות הפונקציה **isEmpty()**.

התחביר הוא כזה :

```
if isempty(request.querystring("x")) then
```

בדיקה זו מבהירה האם הדף עולה בפעם הראשונה או בפעמים הבאות. אם הדף עולה בפעם הראשונה, אין עדיין ערך לשדה (הפונקציה isEmpty() מחזירה true), ואילו לאחר לחיצה על לחצן **Submit** יהיה תמיד ערך למשתנה x (הפונקציה isEmpty() מחזירה false).

אם כן, במידה והמשתנה חסר ערך (כלומר הדף עלה בפעם הראשונה), נציג את הטופס ונסיים את שליחת המשך הדף על ידי Response.end.

```
<%if IsEmpty(Request.QueryString("x")) then%>
    <form>
        Type in your name: <input type="text" name="x" />
        <input type="submit" />
    </form>
    <%
    Response.End
end if%>
```

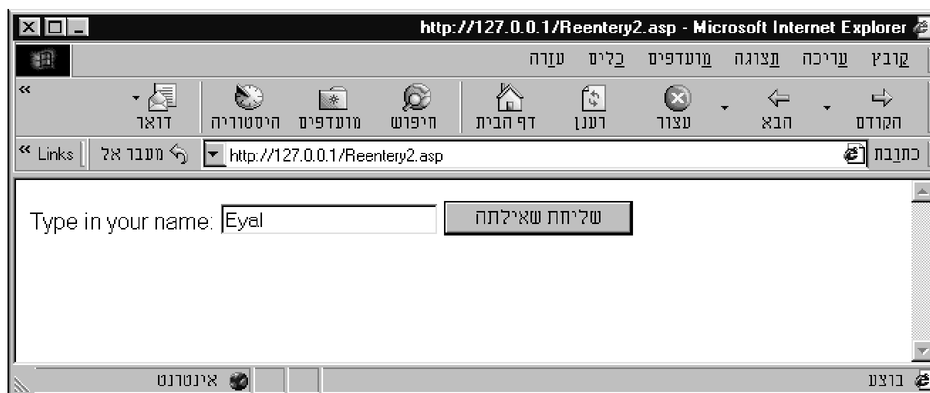
אם התנאי אינו מתקיים, כלומר, יש ערך למשתנה x, נבצע קטע קוד שונה באופן הבא :

```
<body bgcolor="blue" text="yellow">
<font size="5">
<%=Request.QueryString("x")%>,<br />
Welcome to the same page with a different look!</font>
```

להלן הנוסח המלא של קובץ **Reentry2.asp**:

```
<%if IsEmpty(Request.QueryString("x")) then%>
    <form>
        Type in your name: <input type="text" name="x" />
        <input type="submit" />
    </form>
    <%
    Response.End
end if%>
<body bgcolor="blue" text="yellow">
<font size="5">
<%=Request.QueryString("x")%>,<br />
Welcome to the same page with a different look!</font>
```

כך ייראו המסכים בפעם הראשונה והשנייה:





תרשים 25.3

Expires

תכונה זו מגדירה את הזמן שלאחריו. כאשר ינסה המשתמש להיכנס שוב לאותו המסמך, יטען הדפדפן את המסמך מהשרת ולא מהזיכרון המקומי שלו. תכונה זו מקבלת ערכים בדקות. כלומר, אם נקבע שלמסמך מסוים 3 דקות בתכונת Expires, יימחק מסמך זה מזיכרון הדפדפן לאחר 3 דקות.

לדוגמה:

```
<%
response.expires=3
%>
```

ניתן להשתמש גם ב-Expiresabsolute כדי להגדיר תאריך תפוגה מוחלט לדף.

Buffer

תכונה זו, המקבלת ערכי true או false, מגדירה למנוע ASP האם להתחיל ולשלוח פקודות HTML ללקוח לפני ניתוח כל הדף (response.buffer=false), או האם לסיים לנתח את כל המסמך ורק אז לשלוח ללקוח את התוצאה (response.buffer=true). הגדרת **Buffer** חייבת להופיע בתחילת מסמך ASP.

הערך הבסיסי של Buffer הוא False.

AppendToLog

שיטה זו מאפשרת להוסיף שורות טקסט לקבצי LOG של שרת HTTP. ניתן להוסיף עד 80 תווים ללא שימוש בפסיקים. לדוגמה:

```
response.appendtolog("The user left this site after 2 minutes")
```

AddHeader

שיטה זו מאפשרת להוסיף שדות ל-HTTP Header באופן הבא:

```
response.addheader "newfield","valuefornewfield"
```

פרק 26

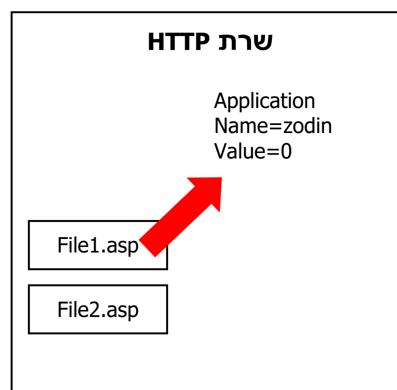
אובייקט Application

לעיתים נוצר הצורך לשמור מידע גלובלי ברמת היישום. הדוגמה הקלאסית לכך, היא הצורך להוסיף לאתר **מונה ביקורים** (counter). מכיון שדפי ASP מתעוררים לחיים עם בקשה מהמשתמש ו"נסגרים" עם סיום ביצוע פעולתם, אין אפשרות לשמור ערכים, כגון מספר המבקרים בדף ASP. לצורך כך, נוצר אובייקט Application המהווה אובייקט חיצוני ליישום המכיל ערכים ונגיש לכל דף ASP בשרת. באובייקט זה ניתן לאחסן ערכים קבועים או משתנים אשר ישמשו את היישום (מכאן נגזר השם Application). לצורך הבנת הרעיון העומד מאחורי השימוש באובייקט זה, נבחן את התהליכים המתבצעים בהוספת מונה ביקורים לאתר.

הוספת מונה ביקורים (counter)

שלב א

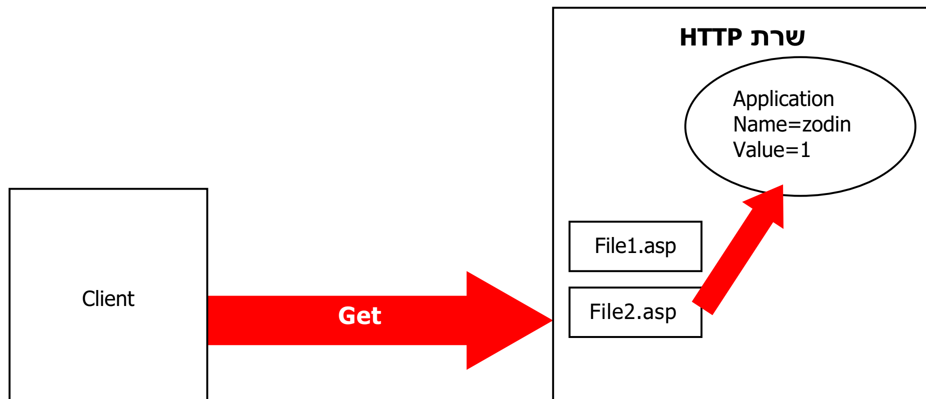
בשלב זה יוצר מסמך ASP אובייקט Application ומעניק לו את השם "zodin". יש צורך בשם עבור כל אובייקט Application כדי לאפשר מספר רב של משתנים מסוג זה. מייד עם יצירת האובייקט וקביעת שמו, קובע מסמך ASP את הערך ההתחלתי של האובייקט כ- 0.



תרשים 26.1

שלב ב

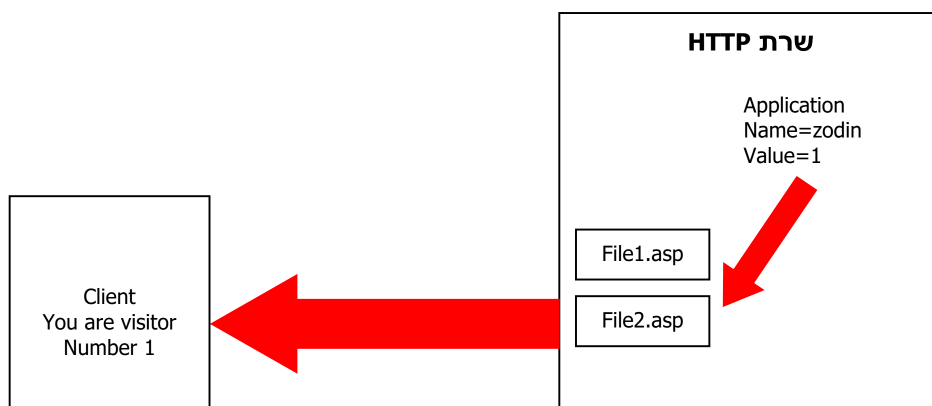
משתמש נכנס לאתר ומבקש את אחד ממסמכי ASP. אותו מסמך ניגש ומעלה את ערכו של אובייקט Application הקרוי zodin ב- 1.



תרשים 26.2

שלב ג

מסמך ASP לוקח את ערכו החדש של zodin ושולח אותו ללקוח (משורשר עם המלל – you are visitor Number 1).



תרשים 26.3

מרגע זה ואילך, כל מסמך ASP יוכל לבצע את אותו תהליך וכך יישמר מספר המבקרים באתר.

אובייקט Application ממשיך להתקיים עד לסגירת השרת או למחיקתו בצורה יישומית.

יצירת אובייקט Application

תחביר יצירת אובייקט **Application** הוא פשוט. יש לכתוב את המילה Application ולאחריה את שם המשתנה אותו ניצור, כגון "zodin". לקביעת ערך המשתנה נוסיף את הסימן **שווה** (=) ולאחריו את הערך הרצוי. לדוגמה:

```
application("zodin") = 1
```

כדי לבצע את קטע קוד זה, נכתוב אותו במסמך ASP ונפנה אליו פעם אחת. כדאי להוסיף מלל כלשהו כדי לקבל אינדיקציה שהקוד התבצע (קובץ **application_create.asp**):

```
<%
application("zodin") = 1
%>
The application variable has been set
```

כדי לראות את ערך המשתנה, נציג אותו באחת משתי הדרכים שנלמדו לשליחת מידע ללקוח:

```
The application variable "zodin" is set to <%=application("zodin")%>
```

או

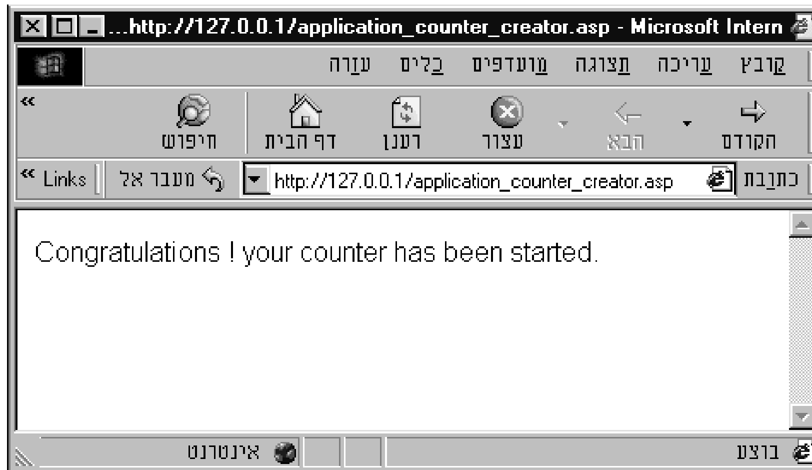
```
<%
response.write "The application variable 'zodin' is set to" & application("zodin")
%>
```

בניית מונה ביקורים באתר

לצורך בניית המונה ואחסונו במשתנה מסוג Application, נבנה מסמך ASP אשר ייצור את המשתנה Excellent_counter וישים בו את הערך אפס (קובץ **application_counter_creator.asp**):

```
<%
application("excellent_counter") = 0
%>
Congratulations ! your counter has been started.
```

נריץ את קובץ ASP פעם אחת ונקבל את התשובה :



תרשים 26.4

כעת, נבנה מסמך ASP אשר יעלה את ערך המשתנה excellent_counter ב- 1 ויציג את ערכו למשתמש. כמו כן, נשרשר את המשפט ": You are visitor number". זהו קובץ counter_page.asp) :

```
<body bgcolor="teal" text="white">
<font size="7">
<%
application("excellent_counter")=application("excellent_counter") + 1
response.write "You are visitor number :" & application("excellent_counter")
%>
</font>
```

נפנה אל המסמך **counter_page.asp** בפעם הראשונה ונקבל את המסך הבא :



תרשים 26.5

כאשר נבצע טעינה מחודשת של המסמך על ידי **רענן** (refresh), נקבל:



תרשים 26.6

בכל פעם שניגש למשתנה excellent_counter, נקבל את הערך שקבע הגולש הקודם.

contents

לאובייקט Application יש אוסף המאגד בתוכו את כל משתני Application שנוצרו. עם זאת, לא ניתן לגשת ל-**Content** באופן רגיל, אלא יש צורך להשתמש בלולאת for each (המוזכרת בהקשר לאובייקט Request).

וכך, עבור כל **פריט** (item) ברשימת האובייקטים מסוג application (application.contents), יוצג ערך המשתנה באופן הבא (קובץ **application_contents.asp**):

```
<%
for each item in application.contents
response.write application.contents(item) & "<br />"
next
%>
```

זאת במידה וקיימים אובייקטים מסוג זה.

Contents.remove()

שיטה זו מאפשרת מחיקת אובייקט application ספציפי, באופן הבא:

```
Application.Contents.Remove("zodin")
```

Contents.removeall()

שיטה זו מאפשרת מחיקת כל האובייקטים מסוג Application, תוך שימוש בתחביר:

```
Application.Contents.RemoveAll()
```

בניית Chat פשוט באמצעות ASP

פרוטוקול HTTP הינו פרוטוקול חסר **קשר רציף** (stateless), ולכן קיימת בעיה עקרונית בבניית chat המחייב קשר רציף לכל המשתתפים בשיחה, מכיון שיש לשלוח עדכונים על משפטים חדשים שהוזנו. עם זאת, ניתן לדמות קשר רציף על ידי מנגנון השהיה הטוען שוב את הדף המעודכן מהשרת כל פרק זמן מסוים. זאת נבצע על ידי פונקציית JavaScript הנקראת **setTimeout()**.

את ערכי המשתנים באובייקט Application יכולים לראות כל המשתמשים, ולכן נוכל להשתמש בו כדי לאחסן את פרוטוקול השיחה העדכני. נגדיר אובייקט Application שיצבור את המשפטים המוזנים מכל המשתתפים בשיחה, יפריד אותם באמצעות
 ויאפשר לכולם לצפות בתוכנו.

נבנה frameset אשר יכיל חלק תחתון סטטי שיאפשר הזנת משפטים, וחלק עליון אשר יכיל את פרוטוקול השיחה ויתעדכן כל פרק זמן נתון (קובץ **ChatFrameSet.html**):

```
<html>
    <head>
        <title>
        </title>
    </head>
    <frameset rows="70%,*" border="1">
        <frame src="chat.asp" name="chatWin" />
        <frame src="input2.asp" name="inputWin" />
    </frameset>
</html>
```

כעת, נבנה את החלק העליון המתעדכן כל 2 שניות ומדפיס את משתנה Chat של אובייקט מסוג Application (המשותף לכל המשתמשים), המכיל את פרוטוקול השיחה (קובץ **Chat.asp**):

```
<script language="javascript">
    function reloadPage()
      {
        location.reload() ;
      }
      setTimeout('reloadPage()',2000) ;
</script>
<%=application("chat")%>
```

השיטה **location.reload()** טוענת את הדף מזיכרון המטמון של הדפדפן. כדי להכריח את הדפדפן לטעון את הדף מהשרת יש לכתוב: location.reload(true).

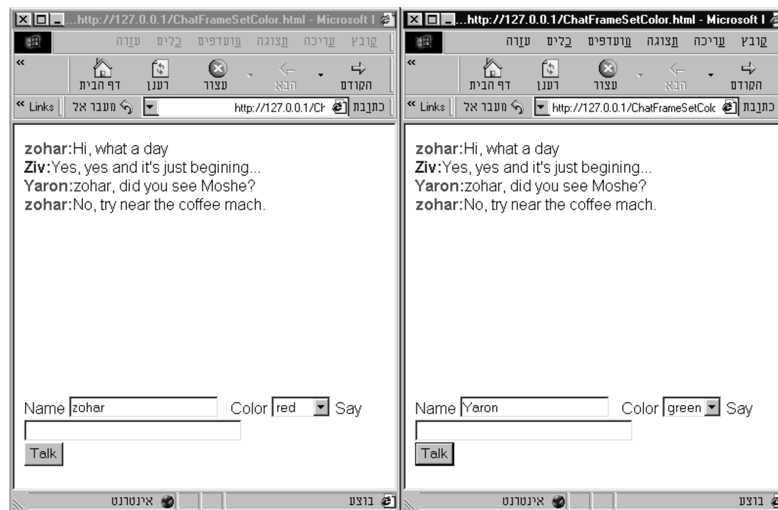
נבנה את החלק התחתון המקבל את שם המשתתף בשיחה ואת המשפטים. חלק זה אף אחראי על הטיפול באובייקט Application. עם כל הזנה של מידע, ישרשר חלק זה שבירת שורה
, שם המשתמש שהזין את המשפט והמשפט עצמו (שים לב כי נשתמש בתפיסת Page Reentry). ראה קובץ **Input.asp**:

```
<%if not(isempty(Request.Form("say"))) then
    myName=Request.Form("myName")
    iSaid = Request.Form("say")
    application("chat")=application("chat") &_
    "<b>" & myName & ":</b>" &_
    iSaid & "<br />"
end if%>
<form method="post">
  Name <input type="text" name="myName" value="<%=myName%>" /><br />
  Say: <input type="text" name="say" size="30" /><br />
  <input type="submit" value="Talk" />
</form>
```

שיפור של chat יכלול צבעים שונים לכל משתתף באופן הבא (קובץ **inputColor.asp**):

```
<%
colorArray = Array("red","green","blue")
if not(isempty(Request.Form("say"))) then
      myName=Request.Form("myName")
      myColor=Request.Form("userColor")
      iSaid = Request.Form("say")
      application("chat")=application("chat") &_
        "<font color=" & myColor & "><b>" &_
        myName& ":</b></font>" & iSaid & "<br />"
end if%>
<form method="post">
Name <input type="text" name="myName"
         value="<%=myName%>" />  
Color <select name="userColor">
          <%for i=0 to 2
                   if colorArray(i)=myColor then%>
                     <option selected><%=colorArray(i)%></option>
                   <%else%>
                     <option><%=colorArray(i)%></option>
                   <%end if%>
          <%next%>
        </select>
Say <input type="text" name="say" size="30" /><br />
<input type="submit" value="Talk" />
</form>
```

כדי לבדוק את ה-Chat הצבעוני יש להפעיל את קובץ **ChatFrameSetColor.html**. ניתן לדמות מספר משתמשים ב-chat על ידי פתיחת שני חלונות (או יותר). התוצאה תיראה כך:



תרשים 26.7

פרק 27

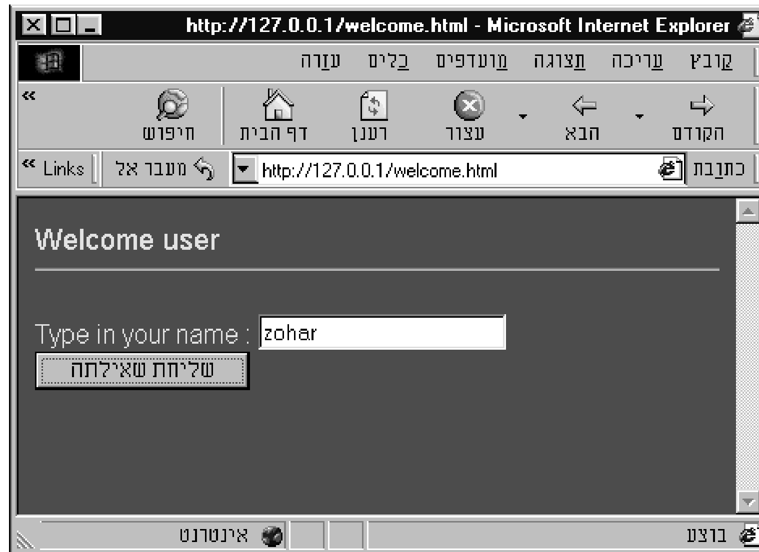
אובייקט Session

האובייקט Application מיועד לשמור נתונים גלובליים ברמת היישום. בפרק זה נראה כיצד ניתן לשמור נתונים שונים עבור כל גולש. ישנם הרבה יישומים לשמירת נתונים עבור כל גולש ואנו נעסוק באחד החשובים בהמשך הפרק, אך תחילה ניישם רעיון פשוט המגדיר כי בכל דף באתר, המשתמש יקבל את שמו (זאת לאחר שהקליד אותו בדף הראשון).

נתחיל בבניית מסמך HTML המכיל טופס להזנת שם משתמש (קובץ **welcome.html**) :

```
<html>
    <head>
        <title>
        </title>
    </head>
    <body bgcolor="red" text="yellow">
        <font size="4">Welcome user</font>
        <hr />
        <form action="session1.asp">
            <font size="3">Type in your name :
            <input type="text" name="n" />
            <br />
            <input type="submit" />
            </font>
        </form>
    </body>
</html>
```

הדף הראשון ייראה כך :



תרשים 27.1

לאחר מכן, נבנה את מסמך ASP אשר יקבל את המשתנה n ויזין אותו אל משתנה מסוג session.

יצירת משתנה מסוג session הינה פשוטה ותואמת את התחביר ליצירת משתנה מסוג application, באופן הבא :

```
session("surfer_name")=.....
```

להלן הקוד (קובץ **session1.asp**) :

```
<%
session("surfer_name")=request.querystring("n")
%>
<body bgcolor="blue" text="white" link="yellow" vlink="yellow">
<font size="6">
Welcome <%=session("surfer_name")%>
<br />
<a href="session2.asp">next page</a>
</font>
</body>
```

שים לב כי קיים בקוד קישור לדף נוסף (**session2.asp**). מכיון ששם המשתמש נשמר באובייקט מסוג session, נוכל לקבל את שמו גם במסמכים נוספים, כגון **session2.asp** כדלהלן (קובץ **session2.asp**):

```
<body bgcolor="yellow">
<font size="6">
Welcome <%=session("surfer_name")%>
</font>
</body>
```

כמובן, כל משתמש יקבל את השם שהזין. ניתן לבדוק זאת על ידי כניסה למסמך משני דפדפנים שונים (זאת משום שאובייקט Session משתמש למעשה ב**עוגיות** (cookies) ושומר את המידע על מספר ה-session אצל הלקוח).

כך ייראו שני המסכים של **session1.asp** ו-**session2.asp**:



תרשים 27.2



תרשים 27.3

אם כך, ניתן לשמור מידע עבור כל לקוח. מכיון שהיישום מתבצע בעזרת עוגיות, ניתן לוותר על השימוש באובייקט זה ולקיים את הפונקציונליות שלו תוך שימוש ׳ידני׳ בעוגיות. אך, מכיון שאובייקט Session מקצר תהליכים, מומלץ להשתמש בו במקרים בהם מתעורר צורך שכזה.

אבטחת מידע

אחד השימושים העיקריים של cookies ולכן גם של אובייקט session, הוא אבטחת דפים מסוימים. ניתן לבקש מהגולש להזדהות עבור מסמך, ולכן ניתן להגן על מידע על ידי בדיקת הסיסמה. אולם, בדרך כלל, אנו מעוניינים להגן על מספר רב של דפים, ולכן נוצרת בעיה.

אם נבקש סיסמה בכל דף, הרי שהפכנו את האתר ללא ידידותי בלשון המעטה, ואם נבקש סיסמה רק בדף הראשון המכיל קישורים לדפים נוספים, יוכל כל גולש לבצע גישה ישירה (בה הוא מציין את שם הדף בשורת הכתובת) אל כל דף שאינו מוגן.

לכן, עלינו לאחסן את האישור (במידה וההזדהות עברה בהצלחה) למשתמש באובייקט session, ולבדוק אותו בצורה שקופה בכל דף.

נתחיל בבניית המסמך הראשון, אשר מבקש את שם המשתמש ומבצע בדיקה. אם הבדיקה נכונה, כלומר הוקשה סיסמה נכונה, יוכל המשתמש לראות את תוכן הדף. לצורך כך נשתמש בתפיסת **Page Reentry**. כמו כן, נשתמש בשיטת post אשר אינה מראה את הפרמטרים בשורת הכתובת ומחייבת שימוש בשיטה **request.form()** (קובץ **authentication1.asp**):

```
<%
if isempty(request.form("secret_name")) then
    response.write "<font size='5'>Please enter your secret name:" &_
    "<form method='post'><input type='password' name='secret_name' />" &_
    "<input type='submit' /> </form>"
    response.end
  else
    if request.form("secret_name")="muchraka" then
            session("secret")="ok"
            response.write "<body bgcolor='black' text='white' " &_
            "link='white' vlink='white'>" &_
            "<font size='5'>Welcome " &_
            "<br />You can now access other secure pages like:<br />" &_
            "<a href='authentication2.asp'>secure second page</a>"
      else
            response.write("Wrong password")
    end if
end if
%>
```

אם הסיסמה נכונה, דף ASP מאחסן את האישור ok במשתנה secret מסוג session, ומציג למשתמש את הדף עם הקישור לדף המאובטח הבא.

כעת, מכיון שרק למשתמש אשר הזין את הסיסמה הנכונה (muchraka) יש משתנה מסוג session המכיל את הערך ok, ניתן לבדוק בכל דף מאובטח נוסף כי באמת קיים אובייקט זה והאם ערכו הוא ok באופן הבא (קובץ **authentication2.asp**):

```
<%
if session("secret")="ok" then
   response.write "<font size='5'>Welcome authorized user!"
 else
   response.write "Access denied"
end if
%>
```

ניתן לבדוק את בטיחות האתר על ידי ניסיון גישה ל- **127.0.0.1/authentication2.asp** על ידי דפדפן אחר, או על ידי סגירה ופתיחה של הדפדפן (לצורך מחיקת ה-cookie) ללא מעבר דרך **authentication1.asp**. התשובה שתתקבל היא: Access denied.

Timeout

אובייקט session נוצר עבור כל משתמש ומשמש לתחימת ביקור באתר (session). לכן, יש הגבלת זמן המגדירה כי אם המשתמש לא ביצע כל פעילות במהלך 20 דקות, אובייקט session נמחק. אותן 20 דקות הן ברירת מחדל וניתנות לשינוי על ידי התכונה timeout באופן הבא:

```
Session.Timeout = 12
```

הערכים המתקבלים הם בדקות.

Abandon

כמו כן, ניתן לסיים session באופן יזום על ידי השיטה **abandon** באופן הבא:

```
Session.Abandon
```

עם השימוש בשיטה זו, נמחקים כל האובייקטים המוגדרים כ-session וכל ההגדרות המוגדרות עבור משתמש.

Contents

שיטה זו דומה לשיטה הזהה באובייקט Application, ומאפשרת גישה לכל האובייקטים המוגדרים כ-Session על ידי שימוש בלולאת for each, באופן הבא:

```
for each item in session.contents
  response.write session.contents(item) & "<br />"
next
```

contents.remove()

שיטה זו דומה לשיטה הזהה באובייקט Application. ניתן למחוק אובייקט session מסוים על ידי השימוש בתחביר:

```
session.contents.remove("pooki")
```

contents.removeall()

שיטה זו דומה לשיטה הזהה באובייקט Application ומאפשרת מחיקת כל האובייקטים המוגדרים כ- Session עבור המשתמש, באופן הבא:

```
session.contents.removeall()
```

פרק 28

עוגיות (cookies)

מהן **Cookies**? דפדפן, כמו IE, שומר קובץ מיוחד הנקרא Cookie File במחשב הלקוח. באמצעות קובץ זה מאחסן השרת בו מבקר הגולש מידע אודות העדפות הגולש לאופן הצגת האתר, התוכן המועדף עליו וכדומה. קובץ זה נוצר בשרת ונשמר במחשב הלקוח. לא כל אתר רושם Cookie במחשב הלקוח. קבצי Cookies מסויימים הם זמניים בלבד, אחרים קבועים. למשל, תוקפן של ה-cookies המשמשות למעקב אחר session פג מייד בתום ה-session, לאחר שהמשתמש עוזב את האתר. Cookies אחרות עשויות להישאר במחשב. השרת שרשם cookie אצלך במחשב, יוכל לזהות אותך בפעם הבאה שתבקר באתר. לכן, הוא יוכל לפנות אליך בשמך, לבנות עבורך דפים לפי העדפותיך ובקיצור יוכל לבצע עבורך התאמה אישית.

כדי לאפשר את רישום ה-cookies על ידי הדפדפן יש לוודא שאפשרות זו הופעלה. פתח את תפריט **כלים** (Tools) בדפדפן. בחר מתפריט **אפשרויות** (Internet Options). בחר בכרטיסיה **אבטחה** (Security). בחר בלחצן **רמה מותאמת אישית** (Custom Level). גלול כלפי מטה עד ל-**קבצי Cookies** (Cookies). סמן את שתי האפשרויות **הפוך לזמין** (Enable). לחץ **אישור** (OK).

בפרק זה נבחן את הכלים שמציע ASP לטיפול בסוג זה של אחסון ואחזור נתונים.

ניתן לשתול cookies באמצעות response.cookies ולשלוף באמצעות request.cookies. חשוב לציין, כי למעט מקרים בהם התכונה response.buffer מקבלת ערך True, יש לבצע כל פעילות הקשורה ל-cookies לפני כל סוג של קוד HTML הנשלח לשרת. כלומר, בשורות הראשונות של קובץ ASP.

לכל **Cookie** מבנה ברור המוגדר במחרוזת טקסטואלית ובה :

- שם Cookie אשר לפיו ניתן לגשת אליה.
- ערך Cookie המאפשר אחסון מידע.
- תאריך תפוגת Cookie אשר אם לא קיים, מגביל את קיום ה-Cookie עד סגירת הדפדפן.
- מסלול מורשה הקובע אם מותר גם למסמכים נוספים בשרת לגשת לאותן Cookies.

יש לתת פקודה עבור כל תכונה של Cookie, באופן הבא:

```
Response.cookies("gunga")=12
```

בפקודה זו, הגדרנו Cookie בשם gunga ואחסנו בתוכה את הערך 12.

```
Response.cookies("gunga").expires=date
```

בשורה זו הוגדר כי ל-gunga cookie יינתן תאריך תפוגה.

```
Response.cookies("gunga").path="/"
```

וכמובן, בשורה זו הוגדר Path עבור ה-cookie.

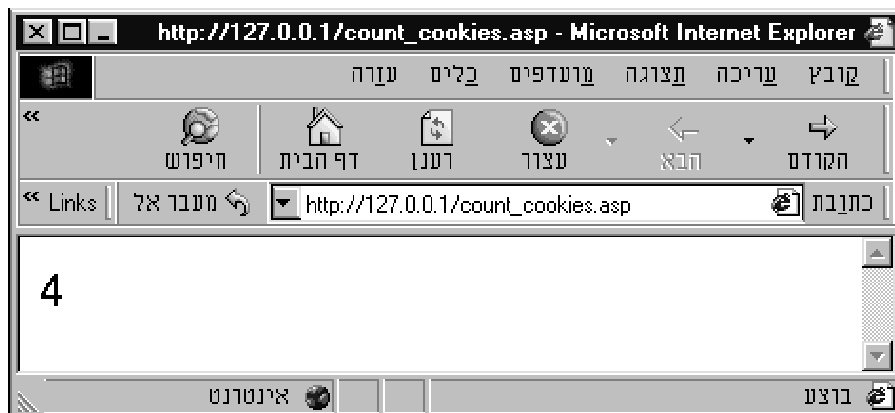
Count

באמצעות התכונה **count**, ניתן לברר את מספר ה-cookies שנרשמו אצל הגולש על ידי היישום הנוכחי.

לדוגמה (קובץ **count_cookies.asp**):

```
<%
response.cookies("a")
response.cookies("b")
response.cookies("v")
response.cookies("f")
response.write "<font size='5'>" & request.cookies.count
%>
```

ולהלן התוצאה:



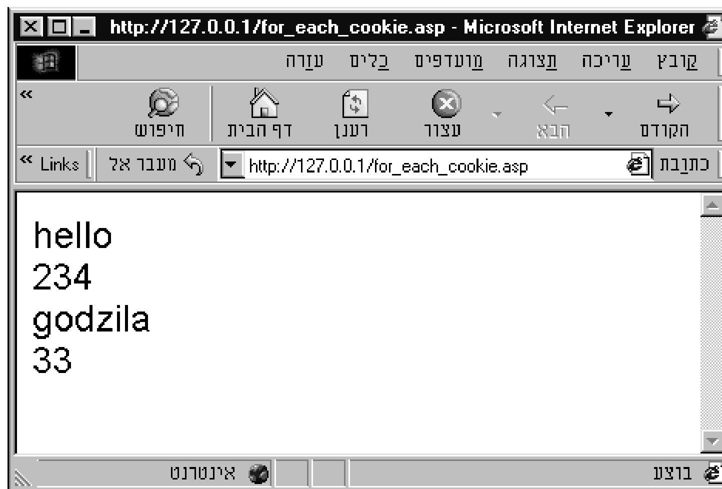
תרשים 28.1

For each

ASP מאפשר מעבר על ה-cookies כמערך על ידי שימוש בלולאה **for each**, באופן הבא (קובץ **for_each_cookie.asp**):

```
<%
response.cookies("a")="hello"
response.cookies("b")=234
response.cookies("v")="godzila"
response.cookies("f")=33
for each item in request.cookies
   response.write "<font size='5'>" & request.cookies(item) & "<br />*"
next
%>
```

התוצאה תיראה כך:



תרשים 28.2

שים לב, כי כל cookie הוצגה בשורה נפרדת, מכיון שהגישה לכל cookie התבצעה כ-request.cookies(item).

התייחסות רק ל-request.cookies תניב תוצאה שונה (קובץ **all_cookies.asp**):

```
<%
response.cookies("a")="hello"
response.cookies("b")=234
response.cookies("v")="godzila"
response.cookies("f")=33
response.write "<font size='5'>" & request.cookies
%>
```

הפעלת קובץ **all_cookies.asp** מציגה את החלון הבא :



תרשים 28.3

מפתח

ניתן לאחסן מספר נתונים רב בתוך cookie על ידי שימוש ב**מפתח** (key). במקום לשמור מספר רב של cookies שונות, ניתן לתת לאותה cookie מפתחות שונים אשר יאפשרו לשלוף את המידע בקלות. לדוגמה, אם נרצה לאחסן בנוחיות פרטי משתמש, כגון : שם פרטי, שם משפחה וטלפון, נגדיר cookie אחת בשם user_details. בתוך cookie זו במבנה של מערך, נגדיר את שאר הנתונים בצורה הבאה :

```
response.cookies("user_details")("firstname")="maori"
response.cookies("user_details")("lastname")="blumenfeld"
response.cookies("user_details")("phonenumber")="02985658"
```

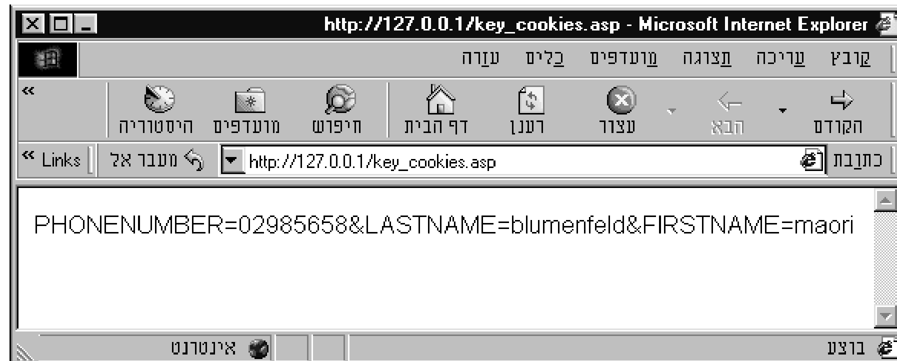
כעת, ניתן לגשת לנתונים בצורה הפשוטה הבאה :

```
response.write request.cookies("user_details")
```

הקוד המלא ייראה כך (קובץ **key_cookies.asp**) :

```
<%
response.cookies("user_details")("firstname")="maori"
response.cookies("user_details")("lastname")="blumenfeld"
response.cookies("user_details")("phonenumber")="02985658"
response.write "<font size='3'>" & request.cookies("user_details")
%>
```

התוצאה תיראה כך :

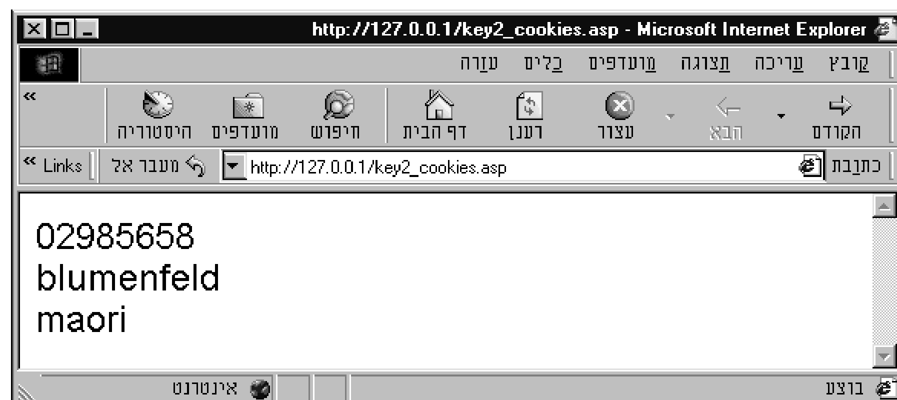


תרשים 28.4

כפי שניתן לראות, קיבלנו את כל ערכי המפתחות כולל שמות המפתח. לא תמיד נרצה לקבל את המידע בצורת מחרוזת אחת, ולכן נוכל גם כאן להשתמש בלולאת for each, באופן הבא (קובץ **key2_cookies.asp**) :

```
<%
response.cookies("user_details")("firstname")="maori"
response.cookies("user_details")("lastname")="blumenfeld"
response.cookies("user_details")("phonenumber")="02985658"
for each item in request.cookies("user_details")
       response.write "<font size='3'>" &_
          request.cookies("user_details")(item) & "<br />"
next
%>
```

התוצאה תיראה כך :



תרשים 28.5

טכניקות לייעול העבודה עם cookies

היתרון היחסי שיש ל-cookies הוא בעובדה שמכיון שהנתונים נשמרים אצל הלקוח, אין צורך במקום בשרת לפרטיהם והעדפותיהם של מיליוני גולשים. עם זאת, הגישה ל-cookies לא תמיד נוחה ולעיתים קל יותר לגשת למשתנים מסוג session. לכן, מומלץ לאחסן את הנתונים אצל הלקוח בפורמט של cookie, אך להעביר את הנתונים למשתנים מסוג session עם התחברות הלקוח. אם נשתמש בדוגמה **key_session.asp** המאחסנת באופן יעיל את פרטי המשתמש ונגדיר כי:

```
response.cookies("user_details")("firstname")="maori"
response.cookies("user_details")("lastname")="blumenfeld"
response.cookies("user_details")("phonenumber")="02985658"
```

נוכל לבנות תסריט קצר המזין את הנתונים הללו לתוך משתנים מסוג session עם הגעתו של הגולש לאתר, באופן הבא (קובץ **Cookies2Session.asp**):

```
<%
session("first_name")=request.cookies("user_details")("firstname")
session("last_name")=request.cookies("user_details")("lastname")
session("phone_number")=request.cookies("user_details")("phonenumber")
for each item in session.contents
   response.write "<font size='3'>" & item & " :  " & _
      session.contents(item) & "<br />"
next
%>
```

מרגע זה, נוכל להשתמש בנוחיות בנתוני Session בכל דף בתחום השימוש של המשתמש הספציפי.

פרק 29

מסדי נתונים

פרק זה הוא פרק עיוני אך חיוני. מכיון שרוב יישומי האינטרנט/אינטראנט כיום מבוססים מסדי נתונים, יש להבין את תפיסת מסד הנתונים ולהכיר את שפת הגישה אליהם. קורא המכיר את שפת SQL רשאי לדלג לפרק הבא, ומי שלא - מוזמן לקרוא פרק זה ולעיין בספר **בסיסי נתונים טבלאיים ושפת SQL - עקרונות ועיצוב** בהוצאת **הוד-עמי**.

כל ארגון צובר נתונים. בין אם זהו בנק הצובר מידע על לקוחותיו, מצב חשבונותיהם והפעילות אותה הם מבצעים, ובין אם זוהי חנות קטנה אשר מאחסנת את נתוני המכירה והעבודה מול הספקים השונים. הצורך באחסון אותם נתונים הוא צורך אמיתי וחשוב, המוגבל על ידי הצורך לשלוף את אותם נתונים במהירות כדי להשתמש בהם בזמן אמת.

למרות עוצמתו הרבה של המחשב המודרני, עדיין קיימת מגבלת זמן עיבוד ונפח איחסון במסדי נתונים גדולים.

אחת הדוגמאות הקלאסיות מדברת על מסד נתונים המאחסן את פרטיהם האישיים של כל תושבי מדינת ישראל. כעת מבקשים ממסד הנתונים לשלוף את כל התושבים אשר שמם הפרטי הוא משה, ועיר מגוריהם היא ראשון לציון. על המחשב לעבור על 6 מיליון רשומות ובכל רשומה לבצע בדיקה ראשונית על שמו הפרטי של התושב ובמקרה של התאמה למשה, לבצע בדיקה נוספת על עיר המגורים. לכל הדעות זוהי פעולה ארוכה ומייגעת.

כעת, מחלקים את המידע במסד הנתונים לטבלאות. כל טבלה מכילה תושבי עיר מסוימת בארץ. לבקשת אחזור מידע דומה, יוכל המחשב לעבור רק על הטבלה המאחסנת את תושבי ראשון לציון.

התוצאה, תהליך בדיקה בודדת (בדיקת השם הפרטי) על כ- 350,000 רשומות בלבד.

ללא ספק, חלוקת המידע לטבלאות ספציפיות מייעלת את תהליך אחזור המידע בצורה משמעותית. אולם, יש דרכים רבות לייעל את תהליכי האחסון והאחזור. לדוגמה, רוב השמות הפרטיים של התושבים יחזרו על עצמם ובכך יוסיפו לזיכרון הדרוש לאחסון המידע. ניתן ליצור טבלה המכילה את כל השמות, אשר ממילא נכפה על מסד הנתונים להכיל, ולהעניק לכל שם מספר. בטבלת התושבים יופיע מספר זה במקום השם.

בדרך כלל, המספר תופס הרבה פחות זיכרון ממחרוזת השם, ולכן השמות עצמם יופיעו פעם אחת כל אחד, וכך נחסך מקום רב בזיכרון.

במקרים כאלה יש לבצע שיקול של חיסכון במקום מול ביצועים, שכן על כל מספר בטבלת התושבים להיבדק מול טבלת השמות.

התהליך בו נבנה מסד הנתונים בצורה היעילה ביותר לאחסון ואחזור נקרא "**נירמול**" (מהמונח Normalization) ודורש התמחות בתחום זה. עם זאת, לבניית יישומים פשוטים המכילים מספר טבלאות וכמות לא גדולה של מידע, אין צורך בהבנה נרחבת, כי אם בכמה כללים בסיסיים.

SQL

יש סוגים רבים של מסדי נתונים המאחסנים מידע בדרכים שונות. מסדי הנתונים אליהם נתייחס בספר זה הם **מסדי נתונים יחסיים** (relational database) המכילים את המידע בטבלאות.

קיימים בשוק מסדי נתונים שנכתבו בשפות שונות, ולכן התעורר צורך לשפה סטנדרטית המאפשרת אחסון ושליפת מידע מכל מסד נתונים. שפה זו היא **SQL** (Structured Query Language).

שפה פשוטה זו מתחלקת לפקודות שליפה, הזנה ועדכון. נתחיל בדוגמה פשוטה.

שליפת נתונים מטבלה

טבלת Phone_book מאחסנת ספר טלפונים קצר:

Phone_book		
First_name	**Last_name**	**Phone_number**
Mooki	Ben-Yaakov	321321
Uri	Hurikan	123456
Starski	Hutch	456578

אם נרצה לשלוף את מספר הטלפון של מוקי בן-יעקב, נכתוב משפט SQL אשר יגדיר:

בחר את **שדה מספר הטלפון** מטבלת **Phone_book** היכן **ששדה השם הפרטי** שווה למוקי

נתקדם עוד צעד לכיוון שפת SQL ונרשום:

בחר את **Phone_number** מטבלת **Phone_book** היכן ש- Mooki=**First_name**

כעת נבחן את משפט SQL:

select **Phone_number** from **Phone_book** where **First_name**='Mooki'

משפט זה יחזיר את התשובה: 321321

אם נרצה לשלוף גם את מספר הטלפון וגם את שם המשפחה של מוקי, נכתוב:

select Pone_number,Last_name from Phone_book where First_name='Mooki'

משפט זה יחזיר את התשובה: 321321 Ben-Yaakov

כמו כן, נוכל לשלוף את כל שורת הנתונים על ידי שימוש בכוכבית (*):

select * from Phone_book where First_name='Mooki'

משפט זה יחזיר את התשובה: Mooki Ben-Yaakov 321321

שליפה אף יכולה להתבצע על פי שני מאפיינים, כגון שם פרטי ושם משפחה:

select * from Phone_book where First_name='Mooki' and Last_name='Ben-Yaakov'

יצירת טבלה

התחביר ליצירת טבלה פשוט אף הוא: צור טבלה בשם Bad_things וקבע את העמודות הבאות (שם מסוג מחרוזת, דירוג מסוג מספר).

משפט SQL ייראה כך:

create table Bad_things (Name varchar, Rating number)

ניתן להבחין כי עמודת Name היא מסוג Varchar. בשל ההבדלים בין מסדי הנתונים, יש לדעת על איזה מסד נתונים אנו עובדים, למרות העובדה שניתן לפנות לכולם ב-SQL, ישנם שינויים קטנים כגון הגדרת סוגי השדות. Varchar מסמל שדה מסוג טקסט במסד נתונים מסוג Access. במסד נתונים מסוג Oracle ניאלץ להשתמש במונח Varchar2.

אם כן, זוהי הטבלה שיצרנו:

Bad_things	
Name	**Rating**

כמובן, עדיין לא קיימים נתונים בטבלה.

הזנת נתונים לטבלה

משפט SQL להזנת נתונים לטבלה ייראה כך:

הזן לטבלת Bad_things לשדות (Name,Rating) את הערכים (Murder,10).

וכך ייראה התחביר האמיתי:

```
insert into Bad_things(Name,Rating) values('Murder',10)
```

התוצאה תתבטא כך:

Bad_things	
Name	**Rating**
Murder	10

ניתן לוותר על שמות השדות אם ההזנה תואמת את מספר וסוגי השדות, לדוגמה:

```
insert into Bad_things values('Theft',4)
```

הפעולה תתבצע ללא הבדל:

Bad_things	
Name	**Rating**
Murder	10
Theft	4

עדכון טבלה

לעיתים עלינו לעדכן מידע הרשום בטבלה. התחביר לעדכון ייראה כך:

```
update Bad_things set Name='Drug Use' where Name='Theft'
```

התוצאה תתבטא כך:

Bad_things	
Name	**Rating**
Murder	10
Drug Use	4

ביטול טבלה

כדי לבטל טבלה השתמש בתחביר הפשוט :

```
drop table Bad_things
```

שפת SQL היא שפה פשוטה בבסיסה, אולם חשוב לציין כי בבניית יישומים אשר אינם פשוטים לחלוטין ומשלבים קשרים בין טבלאות ובדיקות מסובכות, יש להכיר טוב את השפה ולהתנסות בה. המלצת מחברי ספר זה היא, ללמוד על בוריה את השפה מכיון שיישומיה רבים כמעט בכל סביבת פיתוח, לרבות יישומי האינטרנט.

מחיקת רשומות מטבלה

מחיקת רשומות מטבלה מתבצעת באמצעות הפקודה **delete**.

אם נרצה למחוק את השורה הראשונה בטבלה שיצרנו, נרשום :

מחק מטבלה Bad_things היכן שעמודת Name שווה ל-Murder

כך שהתחביר יהיה :

```
Delete from Bad_things where Name='Murder'
```

להעמקה בנושא SQL אנו ממליצים על הספר שיצא בהוצאת **הוד-עמי** :

בסיסי נתונים טבלאיים ושפת SQL - עקרונות ועיצוב

פרק 30

מודל ADO

בפרק זה נלמד כיצד לשמור מידע דרך דפי ASP, וכן, כיצד להציג דפים דינמיים מנתונים השמורים על השרת.

ODBC

כדי להבין כיצד ניגשים מ-ASP למסדי נתונים, יש להסביר תחילה את מהות **ODBC**.

מכיון שמסדי נתונים שונים מתקשרים בשפות שונות, יש צורך בגורם מתווך אשר יידע להסב את שפת SQL המשותפת לכולם, לשפה הייחודית של כל מסד נתונים. גורם זה נקרא **Driver**.

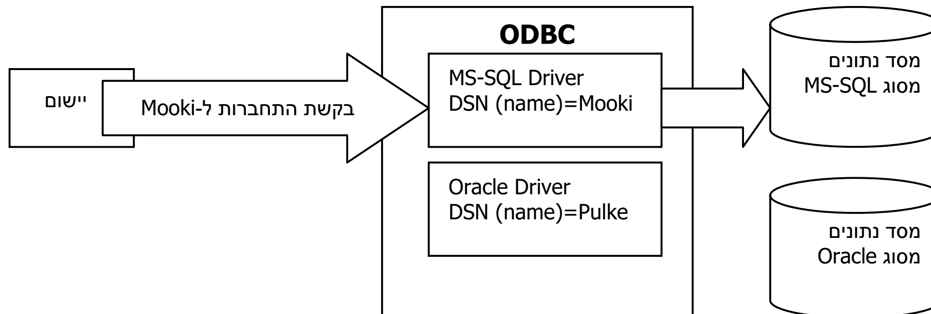
Driver מאפשר להתחבר אל מסד הנתונים ולהעביר לו את משפטי SQL, כך שניתן להתעלם מדרך החיבור, המשתנה ממסד נתונים אחד למשנהו, ומשפת השאילתות, ולבצע חיבור פשוט בשפת SQL תוך התעלמות (חלקית לפחות) מסוג מסד הנתונים.

ODBC בבסיסו הוא קופסת Drivers. ניתן להתקין בו Drivers שונים וליצור חיבורים לוגיים למסד נתונים.

לדוגמה, אם נבחר להתחבר למסד נתונים של Microsoft הנקרא SQL, נוכל לבקש מ-ODBC ליצור חיבור למסד הנתונים בהתבסס על SQL Driver המותקן בו. לחיבור נעניק שם פשוט.

בכל פעם שנרצה בחיבור למסד נתונים, נבקש מ-ODBC את השם הפשוט שהענקנו לחיבור. ODBC ידאג לכל השאר. השם שנעניק לחיבור נקרא **DSN** (Data Source Name).

להלן תרשים המדגים את פעולת **ODBC**:



תרשים 30.1

הסיכוי שכל קוראי הספר משתמשים באותו מסד נתונים הוא קטן, ולכן נתמקד בספר זה במסד נתונים מסוג טקסט.

קבצי טקסט כמסד נתונים

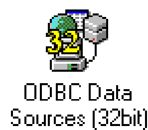
גם ללא תוכנת מסד נתונים, ניתן לאחסן ולאחזר מידע בצורה יעילה. זאת נבצע על גבי קבצי טקסט. Microsoft Text Driver יסייע בידינו להתייחס לקבצי הטקסט כאל טבלאות לכל דבר. מכיון שתפקיד ODBC הוא להפוך את הגישה לנתונים לשקופה למפתח, יעלה בידינו לבנות קוד יישומי מלא.

תהליך הגדרת חיבור למסד נתונים מבוסס קבצי טקסט

פתח את **סייר Windows** (Windows Explorer).

בדיסק **C** בתיקיה **Inetpub**, פתח תיקיה חדשה בשם **jooki** אשר תאחסן את קבצי הטקסט אשר ישמשו אותנו כמסד נתונים.

בחר בתפריט **התחל** (Start), **הגדרות** (Settings), **לוח בקרה** (Control Panel) ולחץ לחיצה כפולה על **ODBC**.



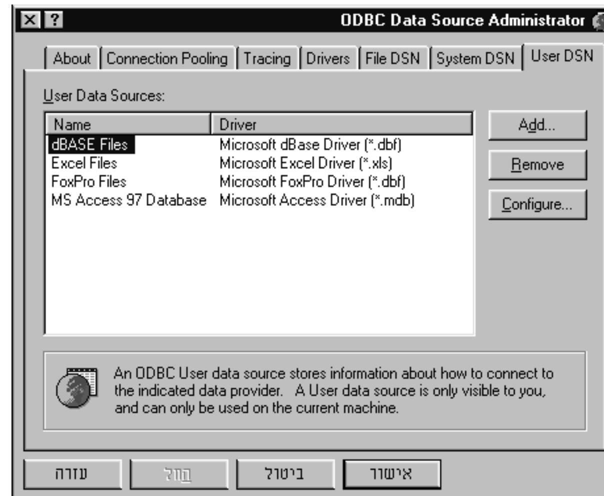
(ייתכן שבמחשבים מסוימים יצוין רק ODBC)

אם אין במחשב את הסמל כלל, יש להיכנס לאתר Microsoft, בכתובת:

www.microsoft.com/data/download.html

ולהוריד את הגירסה האחרונה של ODBC הנקראת: MDAC 2.6 RTM.

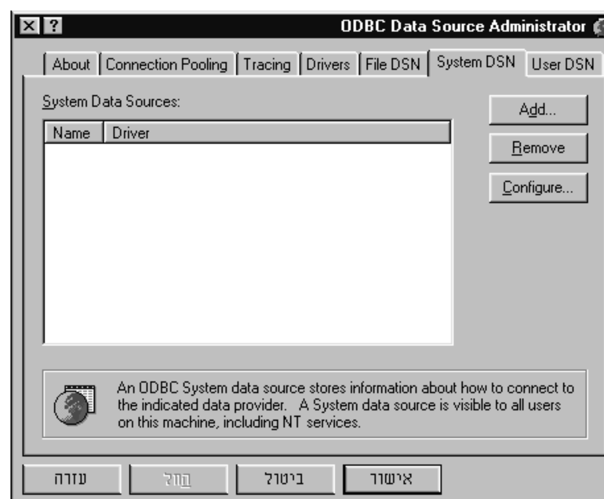
בכל מקרה, מומלץ להוריד את הגירסה העדכנית (נכון ל-March-2001 זהו קובץ של 13MB). זהו המסך שייפתח:



תרשים 30.2

במסך זה ניתן לקבוע קשרים למסדי נתונים ברמת User (הכרטיסיה - user DSN). כלומר, רק המשתמש שמגדיר את הקישור למסד הנתונים יוכל להשתמש בו.

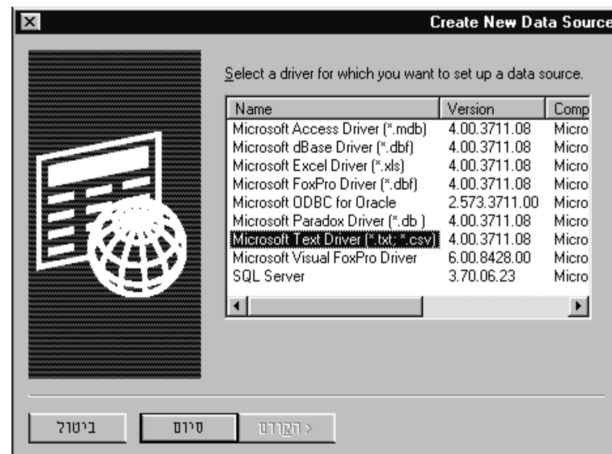
נבחר בלשונית **System DSN** (כזכור, Data Source Name הוא השם הלוגי לחיבור הספציפי) ונקבל את המסך הבא:



תרשים 30.3

מסך זה מאפשר הגדרת חיבור למסד נתונים, המאפשר לכל משתמשי המערכת להתחבר דרכו.

נלחץ על לחצן **Add** (הוסף) ונקבל את המסך הבא :



תרשים 30.4

מסך זה מראה את כל ה-Drivers המותקנים ב-ODBC. נבחר את **Microsoft Text Driver** ונלחץ על **סיום** (Finish).

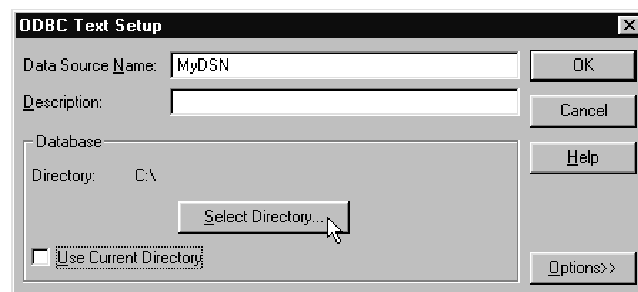
זהו המסך שנקבל :



תרשים 30.5

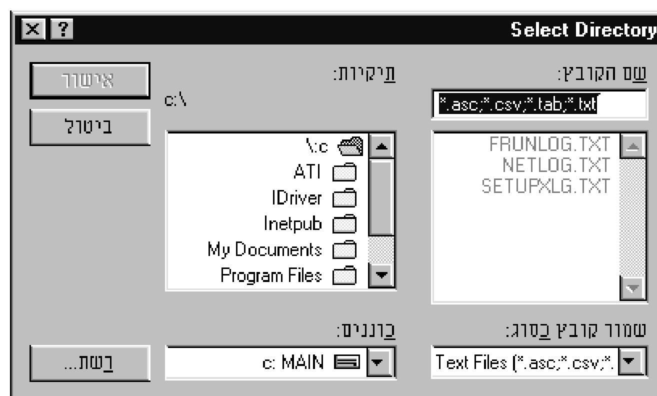
בשדה **Data Source Name** הקלד **MyDSN**. זכור כי כעת נקרא החיבור שלך למסד הנתונים – MyDSN.

בטל את הסימון בתיבת הסימון **Use Current Directory**.



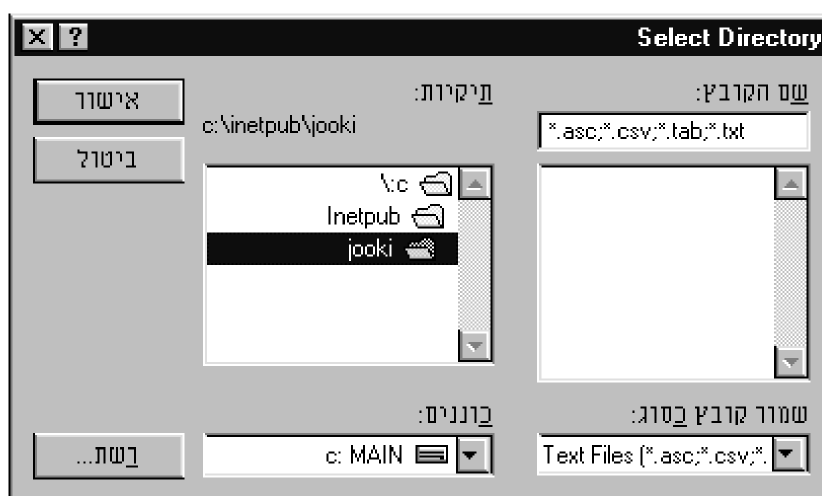
תרשים 30.6

לחץ על הלחצן **Select Directory**,



תרשים 30.7

בחר את התיקיה **jooki** ,



תרשים 30.8

לחץ **אישור** (OK).

לחץ **OK**.

לחץ **אישור** (OK).

כעת אנו מוכנים להתחיל בעבודה.

סיכום ביניים והבהרה

לאחר שביקשנו מ-ODBC להעניק שם לוגי (MyDSN) לחיבור לקבצי הטקסט, לא מעניין אותנו אם המידע נשמר במסד נתונים של Oracle, Microsoft או בקבצי טקסט (שים לב כי עדיין לא הבנו מה הכוונה בשמירת מידע בקבצי טקסט!). אנו מאחסנים ושולפים מידע מ-MyDSN. זוהי גדולתו ופשטותו של ODBC המאפשר לנו לבצע משפטי SQL פשוטים בלי להתחשב באופן בו הנתונים נשמרים בסופו של דבר. למעשה, מרגע שהוגדר DSN, המפתח יכול לשכוח מאופן אחסון הנתונים.

האובייקטים של ADO

אם כן, כל שנותר כעת הוא לשלוח משפטי SQL אל MyDSN כגון יצירת טבלה, שליפת מידע וכדומה. כדי לפנות ל-MyDSN, יש לבצע קריאה מסודרת מ-ASP ל-ODBC. כאן נכנסים לעזרתנו האובייקטים של **ADO**.

ראשי התיבות של **ADO** הם **A**ctive **D**ata **O**bject, אך ייתכן שתיתקל גם ב-**A**ctiveX **D**ata **O**bject.

הרעיון העומד מאחורי ADO הוא לספק מספר אובייקטים מוכנים מראש אשר יאפשרו גישה קלה למסדי נתונים דרך מתווכים, כגון ODBC (ללא האובייקטים המוכנים מראש של ADO, היינו נאלצים לכתוב שורות קוד רבות ומסובכות כדי להתחבר ל- ODBC). האובייקטים בהם נדון בספר זה הם:

Connection – אובייקט המאפשר התחברות קלה אל DSN שהוגדר ב-ODBC.

Recordset – אובייקט שישרת אותנו בשליפת מידע ממסד הנתונים.

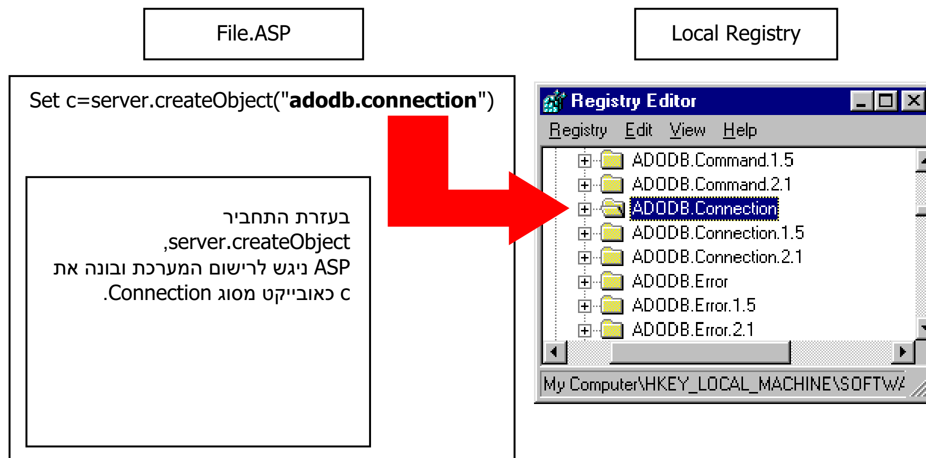
אובייקט Connection

אובייקט זה יבסס עבורנו קשר למסד הנתונים. התחביר ליצירת אובייקט ADO אינו פשוט במבט ראשון, אך הלוגיקה מאחוריו היא פשוטה. התחביר הוא:

```
set c = server.createobject("adodb.connection")
```

המילה set מאפשרת לאתחל את c כאובייקט ולא כמשתנה רגיל.

השימוש באובייקט ובשיטה server.createobject() מורים למנוע ASP לגשת לאובייקטים הרשומים במערכת (אובייקטים אלה מותקנים עם התקנת השרת) ולייצר את האובייקט c על פי התבנית המוכנה שם. במקרה זה, אובייקט Connection.



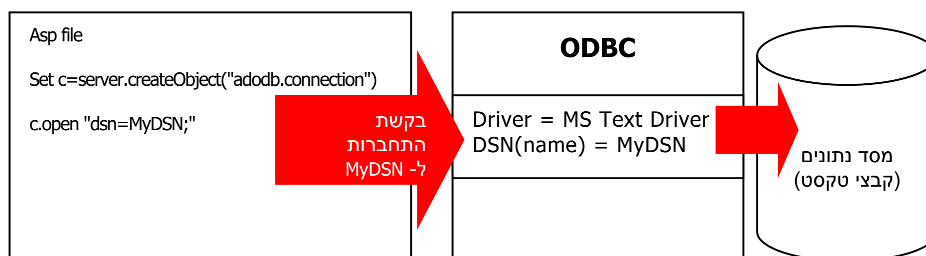
תרשים 30.9

תחביר זה מאתחל את המשתנה c כאובייקט מסוג **Connection**.

כעת, יש להגדיר לאובייקט Connection (שנתנו לו את השם c) לאיזה DSN עליו להתחבר. נבצע זאת על ידי השיטה open של אובייקט Connection.

c.open "dsn=MyDSN;"

התרשים הבא ממחיש כיצד ASP פונה למסד הנתונים דרך ODBC.



תרשים 30.10

בשלב זה, לאחר שקבענו כי c הינו אובייקט מסוג Connection וכי עליו להתחבר ל-DSN שהגדרנו ב-ODBC, ניתן לבצע פקודות SQL על ידי שיטת Execute של אובייקט Connection.

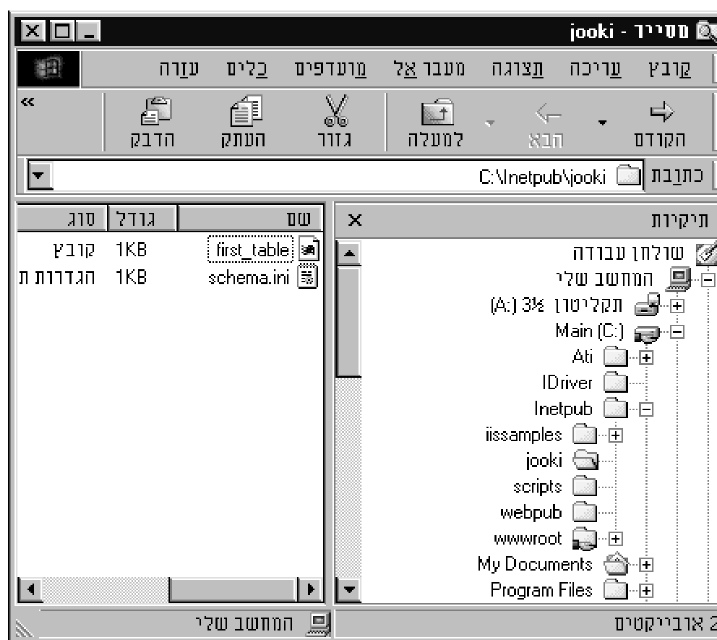
c.execute "create table first_table (Name char, Tel char)"

משפט SQL זה יצר טבלה בשם first_table ובה שתי עמודות. עמודת name מסוג Char (טקסט), ועמודת Tel מסוג Char אף היא.

להלן הקוד המלא (קובץ **table_creation.asp**) :

```
<%
set c = server.createobject("adodb.connection")
c.open "dsn=MyDSN;"
c.execute "create table first_table (Name char, Tel char)"
%>
Your table has been created
```

הרץ קוד זה. לאחר מכן הסתכל בתיקיית jooki החדשה ותגלה כי נוצרו שם שני קבצים : קובץ בשם first_table, וקובץ בשם schema.ini.



תרשים 30.11

אם תפתח את הקובץ first_table בעזרת **פנקס הרשימות** (notepad), תגלה כי זו למעשה הטבלה שיצרת. אל תשנה בקובץ זה דבר.

כך ייראה הקובץ first_table :



תרשים 30.12

כפי שבוודאי ניחשת, הנתונים בטבלה מוקפים בגרשיים ומופרדים בפסיק. ניתן לשרטט את הטבלה החדשה שיצרנו כך :

First_table	
Name	**Tel**

אם כן, יצרנו טבלה על השרת דרך קובץ ASP. כל שנותר הוא להזין מידע אל הטבלה.

הזנת נתונים לטבלה

כעת, נכין קובץ ASP חדש אשר יזין נתונים אל הטבלה החדשה שיצרנו. ניצור אובייקט connection כפי שביצענו בתרגיל הקודם, על ידי שימוש בשיטה ()server.createobject.

```
set c = server.createobject("adodb.connection")
```

נגדיר שוב את DSN, אליו אנו מתחברים, על ידי השיטה open :

```
c.open "dsn=MyDSN;"
```

והפעם נבצע insert אל הטבלה. נבצע זאת מספר פעמים :

```
c.execute "insert into first_table values('mooki','1234')"
c.execute "insert into first_table values('joe','4321')"
c.execute "insert into first_table values('cabuto','6690')"
```

חידוש נוסף בקוד זה, הפעם נסגור את אובייקט connection על ידי שימוש בשיטה Close כדי לא להכביד על מסד הנתונים. אמנם במקרה זה אין אנו מתחברים למסד נתונים, אך מומלץ לסגור את Connection עם סיום הפעולה כדי למנוע מצב בו משתמשים רבים משאירים Connections פתוחים וסותמים את מסד הנתונים.

```
c.close
```

אם נרצה לבטל לחלוטין את אובייקט Connection כדי לחסוך במשאבי מערכת, נשתמש בתחביר:

```
set c = nothing
```

להלן הקוד המלא (קובץ **table_insert.asp**):

```
<%
set c = server.createobject("adodb.connection")
c.open "dsn=MyDSN;"
c.execute "insert into first_table values('mooki','1234')"
c.execute "insert into first_table values('joe','4321')"
c.execute "insert into first_table values('cabuto','6690')"
c.close
set c = nothing
%>
The information was submitted to the database
```

לאחר הרצת הקוד בהצלחה, נפתח שוב את הקובץ first_table ונראה כי המידע אכן נכנס לטבלה:

תרשים 30.13

כמובן כי לצרכי הבהרה ניתן לשרטט טבלה זו כך :

First_table	
Name	**Tel**
mooki	1234
joe	4321
cabuto	6690

הכנת רשימת אורחים באתר

היישום הבא יאפשר למבקרים באתר להירשם ברשימת האורחים. ביישום זה נעשה שימוש באותו DSN שיצרנו (MyDSN).

המטרה היא לבקש את פרטיו האישיים של מבקר ולרשום אותם בטבלה מבוססת קבצי טקסט.

נתחיל במסמך ASP אשר יבנה את הטבלה. נבסס את c כאובייקט Connection :

```
set c = server.createobject("adodb.connection")
```

נחבר את אובייקט Connection שלנו ל-DSN המוגדר ב-ODBC, הרי הוא MyDSN :

```
c.open "dsn=MyDSN;"
```

וניצור את הטבלה guest_book המכילה שם פרטי, שם משפחה ודואר אלקטרוני. כל השדות האלו יהיו מסוג char. כלומר, יכילו מחרוזת טקסט שאינה עולה על 255 תווים.

```
c.execute "create table guest_book(fname char,lname char,email char)"
```

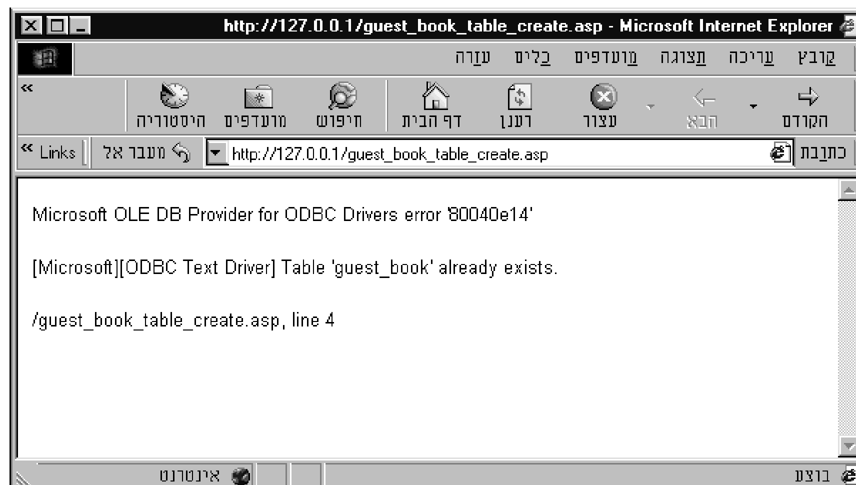
נסגור את אובייקט Conection ונבטל אותו מזיכרון היישום :

```
c.close
set c = nothing
```

לפניך הקוד המלא, כולל מלל HTML אשר ייתן לנו אינדיקציה כי הקוד התבצע (קובץ **guest_book_table_create.asp**) :

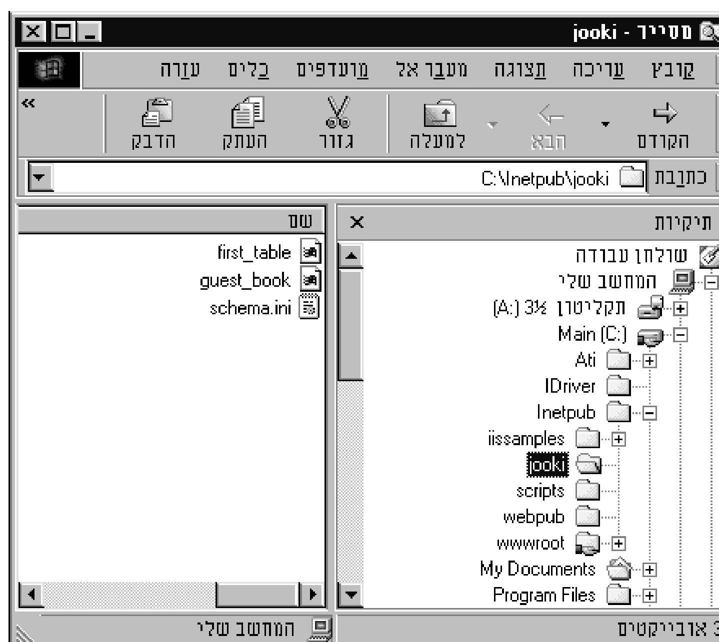
```
<%
set c = server.createobject("adodb.connection")
c.open "dsn=MyDSN;"
c.execute "create table guest_book(fname char,lname char,email char)"
c.close
set c = nothing
%>
Guest_book table has been created successfully!
```

נריץ קוד זה פעם אחת. בתיקיה jooki ייפתח קובץ טקסט בשם guest_book. אם נריץ קוד זה יותר מפעם אחת עלולה להתעורר בעיה, מכיון שהטבלה אותה מנסה קוד זה ליצור כבר קיימת. זו הודעת השגיאה העלולה להופיע אם נריץ קוד זה יותר מפעם אחת:



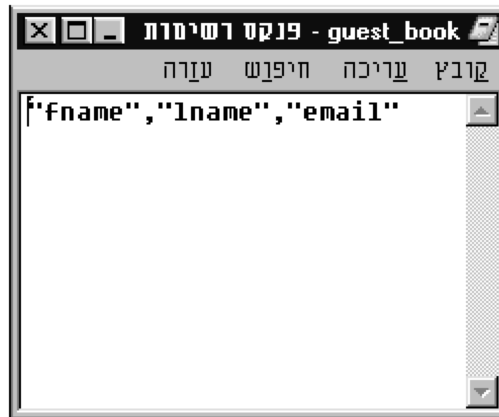
תרשים 30.14

לאחר הרצת הקובץ פעם אחת בלבד, נראה כי בתיקיית jooki נוצר הקובץ guest_book.



תרשים 30.15

אם נפתח את הקובץ guest_book בעזרת **פנקס הרשימות** (notepad), נוכל לראות את מבנה הטבלה :



תרשים 30.16

כעת, נבנה מסמך HTML המכיל את טופס קבלת הפרטים (קובץ **personal_details.html**) :

```
<html>
   <head>
     <title>
     </title>
   </head>
   <body>
     <div align="center">
        <font size="5">Welcome to our Guest Book</font>
        <hr />
        <form action="personal_details_insert_into_guest_book.asp">
          First Name :<input type="text" name="fn" />
            <br />
        Last Name :<input type="text" name="ln" />
            <br />
        E-Mail :<input type="text" name="em" />
            <br />
        <input type="submit" value="Register" />
        </form>
     </div>
   </body>
</html>
```

חשוב לציין, כי כל יישום אינטרנט בנוי מתכנות בצד שרת, כגון ASP שאנו בונים, ותכנות בצד לקוח. תכנות בצד הלקוח דורש לשלב בדיקות JavaScript כדי לוודא שהשדות מולאו בפרטים ואם אפשר גם בפרטים נכונים (הסימן @ בדואר אלקטרוני), וכל זאת טרם שליחה.

כך ייראה מסך HTML :



תרשים 30.17

כעת עלינו לבנות מסך ASP, אשר יקבל את הנתונים מהטופס ויזין אותם אל הטבלה.

נבסס שוב אובייקט Connection ונגדיר DSN :

```
set c = server.createobject("adodb.connection")
c.open "dsn=DSN;"
```

כעת נבצע insert תוך שרשור הנתונים שהגיעו מהלקוח :

```
c.execute "insert into guest_book values('" &
request.querystring("fn") & "', '" &_
request.querystring("ln") & "', '" &
request.querystring("em") & "')"
```

שים לב, כי השרשור אינו פשוט, כיון שיש לכתוב את הגרשיים הבודדים (') המחויבים בתחביר SQL ← ('first name','last name','email'), וכן את הגרשיים הכפולים (") המחויבים בתחביר השרשור של ASP ← "insert into..." & request...

לאחר מכן, נסגור ונבטל את אובייקט Connection :

```
c.close
set c = nothing
```

להלן הקוד המלא הכולל גם את מלל HTML שיופיע על המסך עם סיום הפעולה (שם הקובץ הוא כמובן השם המופיע ב-action של הטופס ב-HTML. ראה קובץ **personal_details_insert_into_guest_book.asp**):

```
<%
set c = server.createobject("adodb.connection")
c.open "dsn=MyDSN;"
c.execute "insert into guest_book values('" & request.querystring("fn") & "','" &_
request.querystring("ln") & "','" & request.querystring("em") & "')"
c.close
set c = nothing
%>
You have been registered successfully!
<a href="personal_details.html">Back to form</a>
```

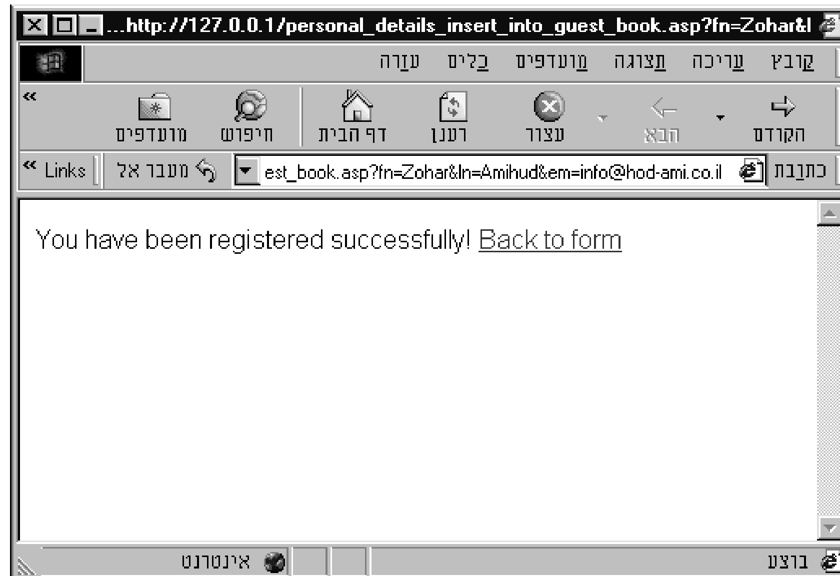
> **הערה**
>
> לקובץ **personal_details_insert_into_guest_book.asp** אין לפנות ישירות, כיון שאז לא יישלחו אליו הנתונים fn,ln,em ותתבצע הזנה ריקה לטבלה! ניתן לפתור בעיה זו על ידי בדיקת isempty פשוטה. כלומר, להתנות את ביצוע הקוד בתנאי:
>
> if isempty(request.querystring("fn")) then
>
> ולא לשכוח end if בסוף!

כעת נעקוב אחר צילומי המסך של היישום. נתחיל במסך HTML:



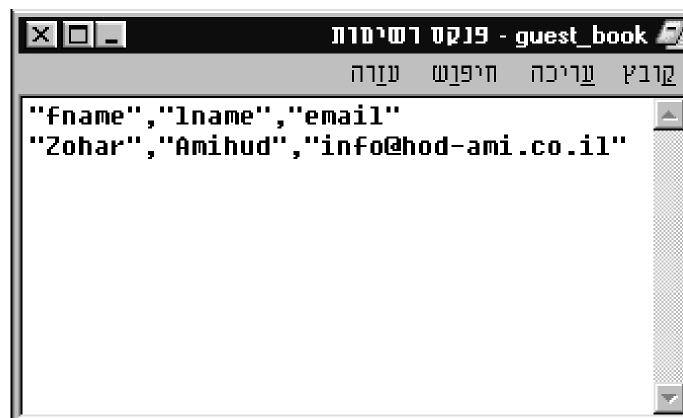
תרשים 30.18

לאחר מילוי הנתונים נקבל:



תרשים 30.19

כעת, נותר רק לפתוח שוב את קובץ guest_book שבתיקיה jooki ולראות כי הנתונים הוזנו:



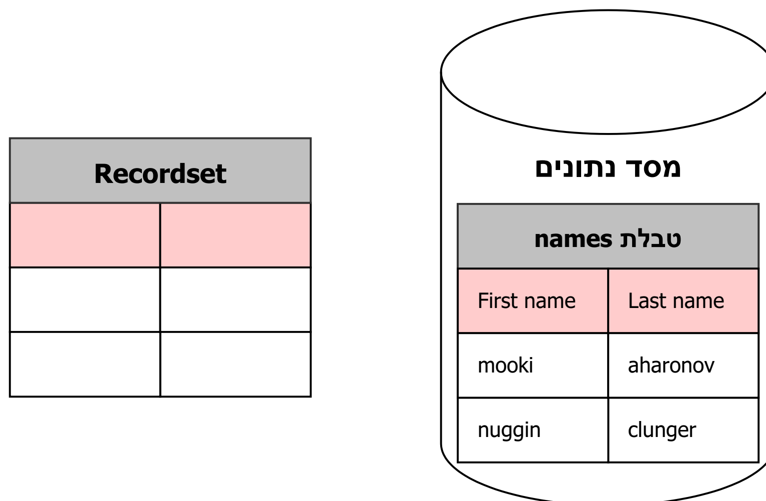
תרשים 30.20

שליפת נתונים והצגתם ב- Web

כדי לשלוף נתונים ממסד נתונים ולהציגם, יש להכיר אובייקט נוסף של ADO הנקרא **Recordset**.

Recordset

לצרכי הבנה, אם נשלוף נתון אחד ממסד נתונים, כגון שם פרטי של אדם, לא תהיה בעיה לאחסן אותו נתון בתוך משתנה רגיל. לעומת זאת, אם נרצה לשלוף את כל הנתונים מטבלה, לא נוכל לאחסן את המידע במשתנה רגיל. אם כך, אנו זקוקים למשתנה שיכול להכיל מידע טבלאי. זהו אובייקט **Recordset**. אובייקט Recordset הוא משתנה טבלאי המעניק לנו כלים נוחים לגשת למידע. הבה נדמה תהליך שליפת מידע מטבלה במסד הנתונים לתוך אובייקט Recordset.

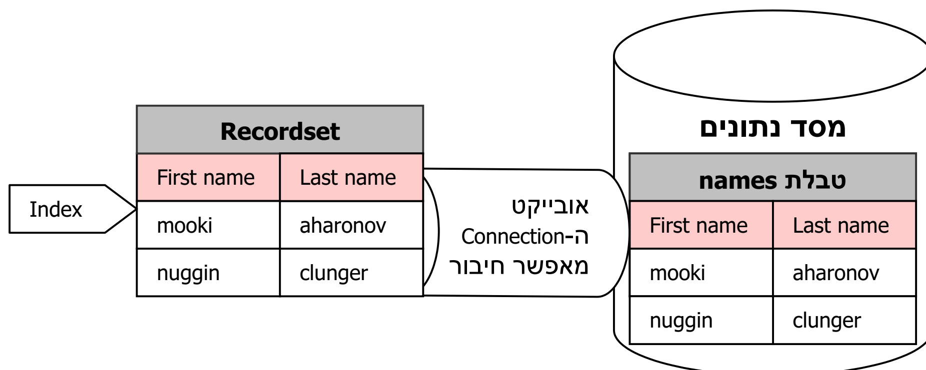


שלב א

תרשים 30.21

בשלב זה, לאחר שנוצר אובייקט Recordset, הוא ממתין ריק. כדי לאפשר ל-Recordset לשלוף את המידע מטבלת מסד הנתונים, יש להגדיר לו כיצד להתחבר דרך אובייקט Connection. לאחר ההתחברות, Recordset שולף את המידע מהטבלה ומאחסן אותה אצלו, כולל שמות העמודות.

שלב ב



תרשים 30.22

בשלב זה, אובייקט Recordset לוקח את הנתונים מטבלת names ומאחסן אותם בצורת טבלה. כיון שאובייקט Recordset נמצא במסמך ASP, ניתן כעת לגשת למידע דרך קוד ASP.

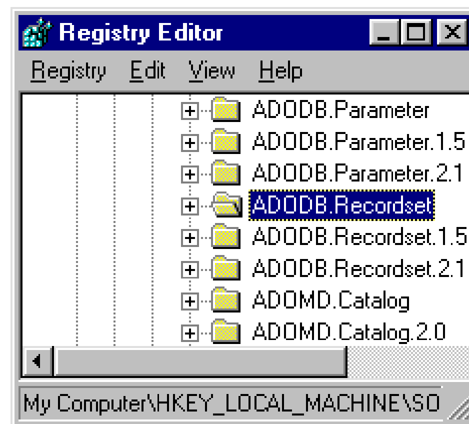
שים לב שבתרשים נוסף אלמנט בשם Index. אלמנט זה הוא אלמנט לוגי המצביע על שורה בטבלה. כפי שמופיע בתרשים, Index מתחיל בשורה הראשונה, כך שאם נבקש מ-Recordset את עמודת First Name, נקבל את mooki. אם נרצה לקבל מידע משורות אחרות, ניאלץ להזיז את Index שורה אחת קדימה ולבקש שוב את עמודת First name, ואז נקבל את nuggin.

תחביר Recordset

יצירת אובייקט **Recordset** זהה ליצירת אובייקט Connection :

```
set r = server.createobject("adodb.recordset")
```

השימוש בתחביר ()server.createobject, מורה למנוע ASP לגשת לרישום המערכת ולאתחל אובייקט על פי אובייקט Recordset הרשום שם :



תרשים 30.23

השלב הבא הוא הגדרת אובייקט Connection אשר דרכו ימשוך אובייקט Recordest את המידע ממסד הנתונים. אם אתחלנו אובייקט Connection בשם c, נשתמש בתכונה activeconnection של אובייקט Recordset כדי להגדיר מול איזה Connection נעבוד:

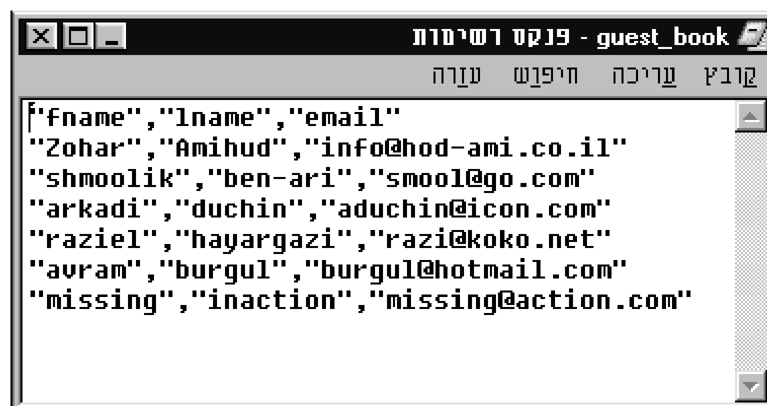
```
r.activeconnection = c
```

השלב הבא, יהיה שליפת המידע מטבלת guest_book אשר הגדרנו בתרגילים הקודמים. שליפת המידע מתבצעת באמצעות התכונה open של אובייקט Recordset:

```
r.open "select * from guest_book"
```

כעת, מכיל אובייקט Recordset (לו קראנו בשם r) את כל המידע מהטבלה.

בטרם נמשיך, יש להיכנס ל- 127.0.0.1/**personal_details.html** ולהזין עוד מספר שמות לטבלה כך שנקבל מספר שורות, כגון:



תרשים 30.24

כעת, נחלץ את המידע מהשורה הראשונה ונציג אותה בדף Web.

זהו קטע הקוד שכתבנו עד כה:

```
set c = server.createobject("adodb.connection") 'create a connection object
c.open "dsn=MyDSN;"
set r = server.createobject("adodb.recordset") 'create a Recordset object
r.activeconnection = c
r.open "select * from guest_book"
```

כיון ש-Index נמצא בשורה הראשונה, כל שעלינו לעשות הוא לבקש מאובייקט Recordset להציג את ערך העמודות fname, lname, email. גישה לעמודה ב-Recordset מתבצעת באמצעות השיטה ()fields, באופן הבא:

```
r.fields("fname")
```

אם כן, נציג למשתמש את התוצאה:

```
response.write "First name=" & r.fields("fname") & "<br />"
response.write "Last name=" & r.fields("lname") & "<br />"
response.write "E-Mail=" & r.fields("email") & "<br />"
```

הקוד המלא ייראה כך (קובץ **one_row_presentation.asp**):

```
<%
set c = server.createobject("adodb.connection")  'create a connection object
c.open "dsn=MyDSN;"
set r = server.createobject("adodb.recordset")   'create a Recordset object
r.activeconnection = c
r.open "select * from guest_book"
response.write "<font size='5'>"
response.write "First name=" & r.fields("fname") & "<br />"
response.write "Last name=" & r.fields("lname") & "<br />"
response.write "E-Mail=" & r.fields("email") & "<br />"
%>
```

התוצאה תיראה כך:



תרשים 30.25

נהדר! הצגנו מידע ממסד נתונים בפעם הראשונה!

הצגת טבלה שלמה

כדי להציג טבלה שלמה, נדרש להזיז את Index שורה אחת קדימה ובכל תזוזה להציג את נתוני השורה בה הוא נמצא. הדרך הבטוחה ביותר לבצע זאת היא באמצעות לולאת do until.

כאשר מתמלא אובייקט Recordset בפעם הראשונה, נמצא Index בתחילת Recordset, כלומר בשורה הראשונה. מקום זה נקרא BOF, כלומר, Beginning Of File. ניתן להתייחס אליו כאל r.bof.

השורה האחרונה מוגדרת כ-End Of File, וניתן להתייחס אליה כאל r.eof.

כעת, ניצור לולאה אשר תזיז את Index שורה אחת קדימה בכל ריצה, עד אשר יגיע ל- r.eof.

תחביר פתיחת הלולאה יהיה:

```
do until r.eof
```

בכל ריצה של הלולאה נציג את נתוני השורה:

```
response.write r.fields("fname") & "---" &
r.fields("lname") & "---" &_
r.fields("email") & "<br />"
```

ונזיז את Index שורה אחת קדימה על ידי שימוש בשיטה movenext של אובייקט Recordset.

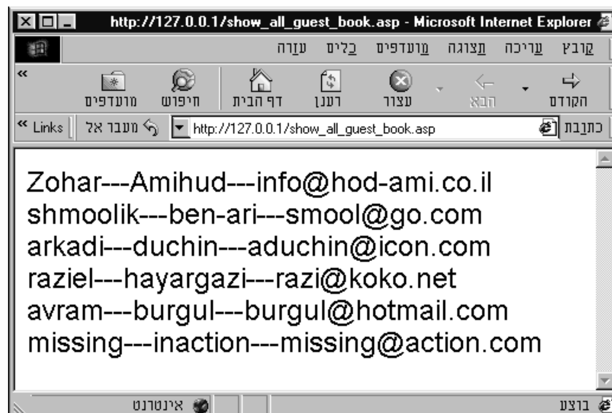
```
r.movenext
```

לאחר מכן, נסגור את הלולאה על ידי התחביר: loop.

כך ייראה הקוד המלא להצגת נתוני כל הטבלה (קובץ **show_all_guest_book.asp**):

```
<%
set c = server.createobject("adodb.connection")
c.open "dsn=MyDSN;"
set r = server.createobject("adodb.recordset")
r.activeconnection = c
r.open "select * from guest_book"
response.write "<font size='5'>"
do until r.eof
   response.write r.fields("fname") & "---" &
r.fields("lname") &_
     "---" & r.fields("email") & "<br />"
   r.movenext
loop
%>
```

וזו תהיה התוצאה:



תרשים 30.26

הצגת הנתונים בטבלת HTML

כדי להציג את הנתונים בטבלת HTML מסודרת, נדרש לשרשר פקודות טבלת HTML אל תוך הקוד בצורה הבאה (קובץ **show_all_guest_book_as_html_table.asp**):

```
<%
set c = server.createobject("adodb.connection")
c.open "dsn=MyDSN;"
set r = server.createobject("adodb.recordset")
r.activeconnection = c
r.open "select * from guest_book"
response.write "<font size='5'>"
response.write "<table border='1'>"        ' this opens a HTML table
response.write "<tr>"  ' this opens a HTML table row
response.write "<th bgColor='teal'><font color='white'>First Name</td>"
response.write "<th bgColor='teal'><font color='white'>Last Name</td>"
response.write "<th bgColor='teal'><font color='white'>E-Mail</td>"
response.write "</tr>"          ' close first table row
do until r.eof
    response.write "<tr>"       ' open a new HTML table row every run of the loop
    response.write "<td>" & r.fields("fname") & "</td>"
    response.write "<td>" & r.fields("lname") & "</td>"
    response.write "<td>" & r.fields("email") & "</td>"
    response.write "</tr>"       ' close the HTML table row every run of the loop
    r.movenext
loop
response.write "</table>"        ' this closes the HTML table
response.write "</font>"
%>
```

יש להקפיד כמובן על פתיחת טבלת HTML לפני תחילת הלולאה, וסגירתה לאחר ביצוע הלולאה. וזו תהיה התוצאה:

First Name	Last Name	E-Mail
Zohar	Amihud	info@hod-ami.co.il
shmoolik	ben-ari	smool@go.com
arkadi	duchin	aduchin@icon.com
raziel	hayargazi	razi@koko.net
avram	burgul	burgul@hotmail.com
missing	inaction	missing@action.com

תרשים 30.27

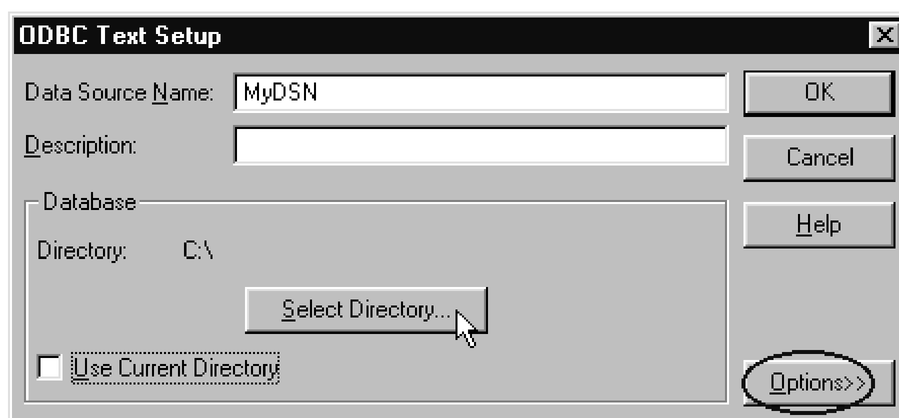
קובץ Schema.ini

אם תפתח את הקובץ **schema.ini** אשר בתיקיית jooki, תגלה כי כאן נמצאות הגדרות הטבלאות. כל שם טבלה מוקף בסוגריים מרובעים ([ ]).

```
[first_table]
ColNameHeader=True
CharacterSet=1255
Format=CSVDelimited
Col1=Name Char Width 255
Col2=Tel Char Width 255
[guest_book]
ColNameHeader=True
CharacterSet=1255
Format=CSVDelimited
Col1=fname Char Width 255
Col2=lname Char Width 255
Col3=email Char Width 255
```

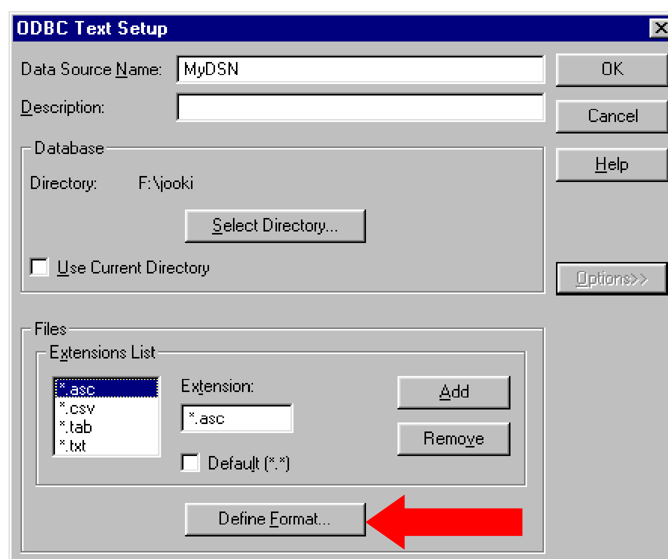
תרשים 30.28

ניתן לראות כי בהגדרות first_table, ישנם שני שדות, Name ו-Tel. שניהם מסוג Char ומכילים עד 255 תווים. ניתן לשנות הגדרות אלו באופן ידני דרך פנקס הרשימות או דרך **Microsoft Text Driver** ב-ODBC, על ידי לחיצה על לחצן **Options** במסך הגדרת DSN.



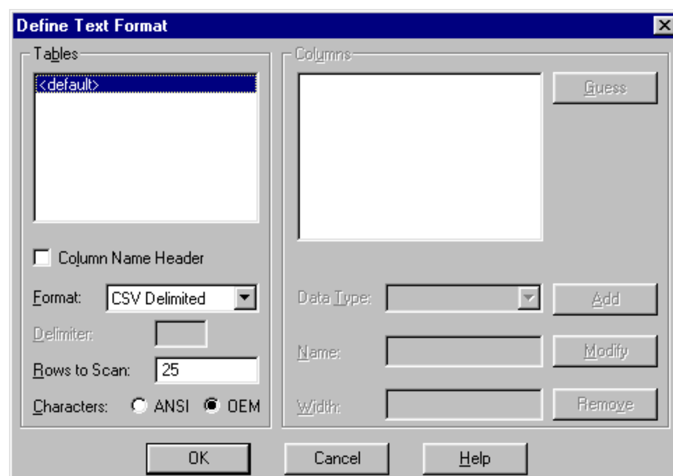
תרשים 30.29

לחיצה על לחצן **Options** תגדיל את המסך ותאפשר ללחוץ על לחצן **Define Format**.



תרשים 30.30

לחיצה על לחצן **Define Format**, תפתח את מסך ההגדרות עבור הטבלאות.



תרשים 30.31

מומלץ בשלב זה לא לשנות הגדרות.

חומר לימוד נוסף בנושא HTML בהוצאת **הוד-עמי**:

ASP 3 ו-Visual InterDev סדנת לימוד, כ- 368 עמודים + תקליטור

ASP 3 המדריך השלם, כ- 928 עמודים + תקליטור

אינדקס עברי

נ

ס

ע

פ

צ

ק

ר

ש

ת

אינדקס לועזי

P

R

S